oprogramowanie
ELWRO 800 Junior

CP/J
Makroasemblery M80, MAC
i programy pomocnicze

elwro

oprogramowanie
ELWRO 800 Junior

CP/J

Makroasemblery M80, MAC

i programy pomocnicze

7110.5022–01 33–0

Wrocław, maj 1987 rok

ZAKŁADY ELEKTRONICZNE "ELWRO"

Wydanie I

7110.5022-01 33-0

Wrocław, maj 1987

Niniejszy podręcznik zawiera opis zastosowań i opis dyrektyw makroasemblerów M80 i MAC, tworzących przesuwalne kody wynikowe dla mikrokomputera ELWRO 800-JUNIOR oraz wykaz instrukcji języka wewnętrznego mikroprocesorów /patrz DODATEK A i B/. Wszystkie programy, których opisy znajdują się w podręczniku pracują pod kontrolą systemu operacyjnego CP/J.

Są to programy:

- makroasembler M80,
- makroasembler MAC,
- program pomocniczy CREF80,
- program łączący L80,
- debuggery symboliczne SID i ZSID

Przed przystąpieniem do uruchamiania w/w programów należy zapoznać się z treścią podręcznika "CP/J. SYSTEM OPERACYJNY CP/J dla mikrokomputera ELWRO 800-JUNIOR".

SPIS TREŚCI

1. WSTĘP 1
1.1. Symbol programu 1
1.2. Nazwa programu 1
1.3. Nośnik 1
1.4. Rodzaj komputera 1
1.5. System operacyjny 1
1.6. Wymagana wielkość pamięci operacyjnej 1
1.7. Język programowania 1
2. PRZEZNACZENIE 1
2.1. Dziedzina zastosowania 1
2.2. Krótki opis rozwiązywanych zadań 1
3. OGÓLNA CHARAKTERYSTYKA PROGRAMU 1
4. DANE WPROWADZANE 2
5. DANE WYPROWADZANE 2
5.1. Format listingu 2
5.2. Format tablicy symboli 5
5.3. Listing z opisem odwołań 6
6. OBSŁUGA OPERATORSKA I KOMUNIKATY DLA UŻYTKOWNIKÓW 6
6.1. Sposób wywoływania i sterowania pracą makroasemblera 6
6.2. Sygnalizacja błędów 9
6.2.1. Sposób sygnalizacji błędów 9
6.2.2. Kody błędów w listingu 10
6.2.3. Informacje o błędach na końcu listingu 12
6.3. Sposób wywoływania i sterowania pracą programu .. CREF80 13
7. POSTAĆ PROGRAMU ŹRÓDŁOWEGO I DYREKTYWY 14
7.1. Format programów źródłowych 14
7.1.1. Uwagi ogólne 14
7.1.2. Instrukcje w języku makroasemblera 15

7.1.3. Nazwy 16
7.1.4. Stałe numeryczne 17
7.1.5. Stałe znakowe 18
7.1.6. Łańcuchy znaków 19
7.2. Wyznaczanie wartości wyrażeń 19
7.2.1. Operatory arytmetyczne i logiczne 19
7.2.2. Tryby wyrażeń 22
7.2.3. Symbole zewnętrzne 24
7.2.4. Kody operacji jako operandy 24
7.3. Dyrektywy w programie źródłowym 25
7.3.1. Dyrektywy wyboru zestawu instrukcji 25
7.3.1.1. .Z80 - instrukcje mikroprocesora Z80 .. 25
7.3.1.2. .8080 - instrukcje mikroprocesora 8080 . 25
7.3.2. Dyrektywy definiowania danych i symboli . 26
7.3.2.1. DB - definiowanie bajtów danych 26
7.3.2.2. DC - definiowanie znaków 27
7.3.2.3. DS - definiowanie bajtów pamięci 27
7.3.2.4. DW - definiowanie słów 28
7.3.2.5. EQU - przyrównanie 28
7.3.2.6. EXT/BYTE EXT - zewnętrzny symbol 29
7.3.2.7. GLOBAL - globalny symbol 30
7.3.2.8. SET - ustawianie 30
7.3.3. Dyrektywy trybu asemblacji 31
7.3.3.1. ASEG - absolutny tryb asemblacji 33
7.3.3.2. CSEG - asemblacja w trybie przesuwalnego programu 33
7.3.3.3. DSEG - asemblacja w trybie przesuwalnych danych 34
7.3.3.4. COMMON - asemblacja w trybie COMMON ... 34
7.3.3.5. ORG - ustawianie licznika komórek 35
7.3.3.6. .PHASE/.DEPHASE - relokowanie 36
7.3.4. Dyrektywy dotyczące zbioru z programem źródłowym 37
7.3.4.1. .COMMENT - długi komentarz 37

7.3.4.2. END - koniec programu 37
7.3.4.3. INCLUDE/MACLIB - dołącz plik 38
7.3.4.4. NAME - nazwa modułu 38
7.3.4.5. .RADIX - podstawa liczb 39
7.3.4.6. .REQUEST - moduły związane 40
7.3.5. Dyrektywy sterujące listingiem 40
7.3.5.1. PAGE/ x EJECT - nowa strona 40
7.3.5.2. TITLE - tytuł 41
7.3.5.3. SUBTTL/ $ TITLE - podtytuł 41
7.3.5.4. .LIST i .XLIST - włączenie i wyłączenie listingu 42
7.3.5.5. .PRINTX - wydruk 42
7.3.5.6. .SFCOND - wstrzymanie listingu nieasemblowanych fragmentów 43
7.3.5.7. .LFCOND - listing nieasemblowanych fragmentów 44
7.3.5.8. .TFCOND - przełączenie listowania nieasemblowanych fragmentów 44
7.3.5.9. .XALL - listowanie wszystkich fragmentów poza nieasemblowanymi 45
7.3.5.10. .LALL - pełne listowanie rozwinięć makroinstrukcji i powtarzanych bloków 45
7.3.5.11. .SALL - wstrzymanie listowania rozwinięć .. 45
7.3.5.12. .XCREF i CREF - wyłączenie i włączenie generowania tablicy odwołań 46
7.3.6. Dyrektywy dla makroinstrukcji i powtórzeń bloku 46
7.3.6.1. MACRO i ENDM - definiowanie makroinstrukcji. 47
7.3.6.2. LOCAL - deklaracja symboli wewnętrznych makroinstrukcji 49
7.3.6.3. REPT i ENDM - powtórzenie bloku 49
7.3.6.4. IRP i ENDM - powtórzenie bloku według listy parametrów 50

7.3.6.5. IRPC i ENDM - powtórzenie bloku według łańcucha znaków 51
7.3.6.6. EXITM - wyjście z makroinstrukcji 51
7.3.7. Dyrektywy asemblacji warunkowej 52
7.3.7.1. Dyrektywy badające warunek asemblacji 53
7.3.7.2. ELSE - określenie zamiennego kodu przy asemblacji 55
7.3.7.3. ENDIF/ENDC - koniec bloku warunkowej asemblacji 56
8. MAKROASEMBLER MAC 56

Załącznik A. ZESTAWIENIE DYREKTYW MAKROASEMBLERA M80 61
Załącznik B. ZESTAWY INSTRUKCJI DLA MIKROPROCESORÓW Z80 i 8080 66
Załącznik C. PROGRAM ŁĄCZĄCY-L80 73
Załącznik D. DEBUGGER'Y SYMBOLICZNE SID i ZSID 82

1. WSTĘP

1.1. Symbol programu: M80

1.2. Nazwa programu: CP/J MAKROASEMBLER

1.3. Nośnik: dysk elastyczny

1.4. Rodzaj komputera: ELWRO 800-JUNIOR

1.5. System operacyjny: CP/J

1.6. Wymagana wielkość pamięci operacyjnej: minimum 29 KB, a w tym: ok. 14 KB na program makroasemblera, ok. 5 KB na miejsca robocze makroasemblera, ok. 4 KB na bufory i ok. 6 KB na system operacyjny.

1.7. Język programowania: asembler mikroprocesora INTEL 8080.

2. PRZEZNACZENIE

2.1. Dziedzina zastosowania: opracowanie programów dla mikroprocesorów typu INTEL 8080 i Z80.

2.2. Krótki opis rozwiązywanych zadań: translacja programów zapisanych w języku makroasemblera INTEL 8080 lub Z80 na przesuwalny kod wynikowy.

3. OGÓLNA CHARAKTERYSTYKA PROGRAMU

Makroasembler M80 jest programem produkującym przesuwalny kod wynikowy dla mikrokomputerowych systemów wykorzystujących mikroprocesory INTEL 8080 lub Z80.

Jako pomocniczy program dostarczany razem z makroasemblerem występuje program CREF80 umożliwiający uzyskanie rozszerzonego listingu, zawierającego informacje o odwołaniach wewnątrz asemblowanego modułu.

Makroasembler M80 jako głównego nośnika używa dysków elastycznych. W czasie asemblacji program źródłowy jest pobierany z pliku typu MAC, przesuwalny kod wynikowy jest umieszczony w pliku typu REL, a listing jest wyprowadzany do pliku typu PRN.
Dla uzyskania ostatecznej wersji kodu wynikowego programu należy użyć programu łącząco-ładującego, który scala osobno asemblowane moduły programu tj. programu L80.

4. DANE WPROWADZANE

Program translujący M80 czyta program źródłowy umieszczony na dyskietce, w pliku o dowolnej nazwie i obowiązkowym rozszerzeniu MAC. Czytany plik oprócz instrukcji odpowiadających rozkazom mikroprocesora /INTEL 8080 lub Z80/ zawiera dyrektywy /pseudoinstrukcje/ sterujące pracą programu, podlegającego translacji przez M80.
Szczegóły dotyczące sposobu zapisu programu źródłowego i dyrektyw zostały przedstawione w pkt. 7.

5. DANE WYPROWADZANE

5.1. Format listingu /wydruku/

Każda strona listingu makroasemblera M80 posiada nagłówek umieszczony w dwóch pierwszych wierszach w postaci:

```
TITLE   tekst          M80 z.zz       PAGE  x - y
SUBTTL  tekst
```

gdzie:

- TITLE tekst - stanowi tekst ustalony dyrektywą TITLE w programie źródłowym. Jeśli dyrektywa ta nie była użyta w programie źródłowym, wówczas obszar ten pozostaje niezadrukowany,
- z.zz - jest numerem wersji programu makroasemblera,
- x - jest głównym numerem strony i ulega przyrostowi tylko po napotkaniu znaku nowej strony w programie źródłowym,
- y - jest pomocniczym numerem strony, który jest powiększany, gdy wystąpi dyrektywa PAGE w programie źródłowym lub gdy objętość bieżącej strony jest wypełniona do końca,
- SUBTTL tekst - stanowi tekst ustalony dyrektywą SUBTTL w programie źródłowym. Jeśli dyrektywa ta nie była użyta w programie źródłowym, wówczas obszar ten pozostaje niezadrukowany.

Po nagłówku występuje wiersz pusty a następnie rozpoczyna się tekst listingu.
Format wiersza listingu jest następujący:

błąd   n   xx xxxxm   w   tekst

gdzie:

błąd - reprezentuje jednoliterowy kod błędu. Jeśli błąd nie występuje, pozostaje puste, niezadrukowane miejsce,

n - reprezentuje licznik komórek /licznik przydziału miejsca w pamięci/ Jest to czterocyfrowa liczba szestnastkowa

/standardowo/ lub sześciocyfrowa liczba oktalna. Podstawa liczbowa licznika komórek może być wybierana przez parametr /O lub H/ przy wywoływaniu programu makroasemblera /pkt. 6.1/,

- m - wskazuje tryb asemblacji w następujący sposób:

  , tryb przesuwalnego programu,
  " tryb przesuwalnych danych,
  ! tryb COMMON,
  spacja tryb absolutny,
    zewnętrzny symbol,

- xx i xxxx - reprezentuje kod stanowiący wynik asemblacji, xx reprezentuje wartość jednego bajtu. Po tej wartości występuje zawsze spacja, xxxx reprezentuje wartość dwubajtową z tym, że wartość bardziej znaczącego bajtu jest drukowana pierwsza /odwrotnie niż wartości tych bajtów są umieszczone w pamięci/. Po wartości dwubajtowej jest drukowany wskaźnik trybu asemblacji "m" /opisane już powyżej/,

- w - reprezentuje wskaźnik, skąd pochodzi wiersz programu źródłowego. Jeżeli wynika on z dyrektywy INCLUDE, wówczas w tym miejscu drukowana jest litera C. Jeżeli wiersz ten wynika z rozwinięcia makroinstrukcji /MACRO, REPT, IRP, IRPC/, wówczas w tym miejscu drukowany jest znak +.

  W pozostałych przypadkach miejsce to pozostaje puste /spacja/,

- tekst - reprezentuje pozostałą część wydruku wiersza listingu odpowiadającą programowi źródłowemu.

5.2. Format tablicy symboli

Strony listingu zawierające tablice symboli posiadają taki sam nagłówek jak listing zasadniczy, z wyjątkiem tego, że zamiast numeru strony jest drukowana litera S. Na początku tablicy symboli są drukowane w porządku alfabetycznym nazwy makroinstrukcji a następnie również w porządku alfabetycznym pozostałe symbole.
Po każdym symbolu występuje znak tabulacji i wydruk wartości tego symbolu. Po wartości symbolu jest drukowany jeden z poniższych znaków:

I	dla symboli PUBLIC,
U	dla symboli bez określonej wartości,
C	dla nazwy bloku COMMON /dla określonej nazwy bloku COMMON jest drukowana długość tego bloku w bajtach w postaci liczby heksadecymalnej lub oktalnej/,
	symbol zewnętrzny,
spacja	wartość absolutna,
'	wartość względna w trybie przesuwalnego programu,
"	wartość względna w trybie przesuwalnych danych,
!	wartość względna COMMON.

5.3. Listing z opisem odwołań

Istnieje możliwość uzyskiwania listingu z dodatkowymi danymi, zawierającymi informację o odwołaniach wewnątrz asemblowanego modułu programu źródłowego.
Listing taki uzyskuje się przy wykorzystaniu dodatkowego, obok makroasemblera programu przetwarzającego o nazwie CREF80.
Sposób realizacji tego został opisany w pkt. 6.3.
Rozszerzony listing z opisem odwołań występuje w zbiorach LST. Zawiera on dodatkowo w stosunku do normalnego listingu /przechowywanego w zbiorze typu PRN/:

a/ numerację dla każdej instrukcji programu źródłowego zapisaną na pierwszej pozycji w wierszu listingu,

b/ tablicę odwołań drukowaną na końcu listingu. Tablica ta zawiera symbole uszeregowane w porządku alfabetycznym. Dla każdego z symboli podany jest zestaw numerów wierszy programu źródłowego, w których ten symbol jest używany. Numer wiersza w którym jest definiowana wartość tego symbolu jest dodatkowo wyróżniany znakiem #.

6. OBSŁUGA OPERATORSKA I KOMUNIKATY DLA UŻYTKOWNIKÓW

6.1. Sposób wywoływania i sterowania pracą makroasemblera

Po założeniu dyskietki z programem makroasemblera M80 /do dysku A: lub B:/ jest on wywoływany z konsoli systemowej zleceniem nierezydentnym:

M80

W odpowiedzi makroasembler wypisuje na konsoli znak gwiazdki, wskazujący, że oczekuje on na podanie linii dyrektywy ustalającej sposób pracy makroasemblera.
Pliki, z którymi pracuje makroasembler M80 są wyróżniane przez następujące oznaczenia typów /rozszerzeń/:

MAC - program źródłowy,

PRN - plik zawierający listing,

REL - plik zawierający przesuwalny kod wynikowy,

COM - plik zawierający kod ładowalny /absolutny/,

LST - plik zawierający listing z odwołaniami,

CRF - plik pośredni służący do uzyskania listingu z odwołaniami.

Linia dyrektywy ustalająca sposób pracy makroasemblera ma postać:

wynik, list = źródło parametry

gdzie:

źródło oznacza nazwę pliku zawierającego program źródłowy /typu MAC/,

wynik oznacza nazwę pliku na dysku, w którym będzie umieszczony kod wynikowy po asemblacji, stosowany opcjonalnie. Jeżeli nie zdefiniowano nazwy tego pliku, wówczas ma on nazwę taką jak program źródłowy i jest typu REL,

list oznacza nazwę pliku na dysku, w którym będzie umieszczony listing, stosowany opcjonalnie,

parametry — dodatkowe dane sterujące pracą makroasemblera, zapisywane w postaci pary znaków: /litera/. Parametry są stosowane opcjonalnie w dowolnej ilości. Znaczenie poszczególnych parametrów podano dalej.

W przypadku, jeżeli nie są potrzebne oba pliki zapisane z lewej strony równości /tj. wynik i list/ wówczas można stosować skrócone zapisy zilustrowane poniższymi przykładami:

= WODA - asemblacja zbioru WODA.MAC i utworzenie pliku kodu wynikowego WODA.REL bez listingu,

,= WODA - asemblacja zbioru WODA.MAC bez tworzenia pliku kodów wynikowych i listingu /przydatne dla sprawdzenia błędów/,

,WATA=WODA - asemblacja pliku WODA.MAC i utworzenie pliku WATA.PRN zawierającego listing, bez tworzenia pliku kodów wynikowych.

W dyrektywie ustalającej sposób pracy makroasemblera, w pola oznaczające nazwę plików można też stosować wpisywanie nazwy urządzeń w systemie. Przykładowo, dla uzyskania bezpośrednio wydruku listingu programu z pliku WODA, na drukarce można wprowadzić:

,LST: = WODA

Parametry, które można stosować dla sterowania pracą makroasemblera są następujące:

/O - tworzenie pliku listingu z adresami zapisanymi oktalnie,

/H - tworzenie pliku listingu z adresami zapisanymi heksadecymalnie,

/R - wygenerowanie pliku kodów wynikowych o nazwie takiej, jak program źródłowy,

/L - wygenerowanie pliku zawierającego listing o nazwie takiej, jak program źródłowy,

/C - wygenerowanie specjalnego pliku typu CRF, niezbędnego dla otrzymania listingu z opisem odwołań przy użyciu dodatkowego programu CREF80,

/Z - asemblacja pliku źródłowego zawierającego instrukcje mikroprocesora Z80,

/P - każdy przełącznik /P powoduje przeznaczenie dodatkowych 256 bajtów dla pracy stosu w czasie asemblacji. Parametr należy stosować wtedy, gdy w systemie występują błędy,

/M - inicjacja obszarów pamięci zarezerwowanych dyrektywą DS w taki sposób by komórki zawierały wartości zerowe,

/X - zakaz wyprowadzenia na listing fragmentów programu, które nie są asemblowane ze względu na warunkową asemblację.

Możliwe jest jednoczesne wywoływanie i ustalanie sposobu pracy makroasemblera przez wypisanie w jednym wierszu zarówno dyrektywy wywołującej, jak i sterującej.

Przykładowo:

MRL , = WODA

6.2. Sygnalizacja błędów

6.2.1. Sposób sygnalizacji błędów

Makroasembler generuje w czasie pracy dwa rodzaje komunikatów o błędach:

- błędy programu źródłowego sygnalizowane na listingu w pierwszej kolumnie przy pomocy jednoliterowego kodu błędów,
- błędy w postaci informacji wypisywanej na końcu listingu, związane z asemblacją.

Jeśli listing jest wyprowadzony na konsolę, wówczas błędy te wyprowadzane są na konsolę razem z listingiem. Jeżeli listing nie jest wyprowadzony na konsolę, wówczas na konsolę są wyprowaszane te wiersze listingu, które mają zaznaczone błędy, a następnie jest wypisywana informacja o błędach umieszczona na końcu listingu.

6.2.2. Kody błędów w listingu

Kod błędu	Przyczyna
A	Błąd argumentu dyrektywy. Zły format lub poza zakresem.
C	Błąd zagnieżdrzenia warunkowej asemblacji: ELSE bez IF, ENDIF bez IF, dwa ELSE przy jednym IF, ENDC bez COND.
D	Podwójnie zdefiniowany symbol. Odwołanie do symbolu, który został więcej niż raz zdefiniowany.
E	Błąd dotyczący nielegalnego użycia symbolu zewnętrznego.
M	Wielokrotne definiowanie symbolu. Występuje definiowanie symbolu,

który został poprzednio zdefiniowany.

N	Błędna liczba. Zwykle błędna cyfra, np. cyfra 8 w liczbie oktalnej.
O	Błędny kod operacji lub błędna składnia, ENDM, LOCAL na zewnątrz bloku; SET, EQU lub MACRO bez nazwy, błędna składnia kodu operacji, błędna składnia w wyrażeniu /np. opuszczone nawiasy, opuszczony cudzysłów/.
P	Błąd przebiegu asemblera. Wartość etykiety lub nazwy określonej przez EQU jest różna w kolejnych przebiegach asemblacji,
R	Błąd dotyczący przesuwalności /relokacji/. Nielegalne użycie przesuwalności w wyrażeniu, jak np. absolutna wartość - przesuwalna wartość.
U	Niezdefiniowany symbol. Symbol użyty w wyrażeniu jest niezdefiniowany. Dla niektórych dyrektyw w pierwszym przebiegu asemblacji jest drukowany błąd V, a błąd U dla drugiego przebiegu.
V	Błąd wartości. Dyrektywa, która powinna dysponować znaną wartością symbolu w pierwszym przebiegu asemblacji /np. RADIX, .PAGE, DS, IF, IFE/ nie ma zdefiniowanej tej wartości.

Jeśli symbol ten zostanie zdefiniowany później w programie, to przy drugim przebiegu asemblacji nie wystąpi błąd U'.

6.2.3. Informacje o błędach wyprowadzone na końcu listingu

% BRAK KOMENDY END

Brak dyrektywy END - dyrektywa opuszczona lub nie analizowana przez makroasembler, np. ze względu na występowanie we fragmencie programu nie podlegającym asemblowanym przy asemblacji warunkowej.

NIEPRZERWANE ASSEM. WAR.

Co najmniej jeden wskaźnik warunkowej asemblacji nie został zamknięty po dojściu do końca programu źródłowego.

NIEPRZERWANE REPT/IRP/MACRO

Co najmniej jeden w wyżej wymienionych bloków nie ma zakończenia.

ZBYT DUŻO ETYKIET

Tablica symboli budowana w czasie asemblacji wyczerpała posiadający zasób pamięci. Najczęściej jest to związane z użyciem dużej liczby makrodefinicji. W pamięci są przechowywane makrodefinicje łącznie z komentarzami. Aby oszczędzić pamięć można komentarze w makrodefinicjach poprzedzać znakami ";;".

-- BŁEDY /ów/ -- OSTRZEŻENIA (EŃ)

Jest to ostatni wydruk makroasemblera po zakończeniu asemblacji.

Podawana jest łączna liczba wykrytych bezwzględnych błędów /--/ oraz ostrzeżeń, tj. błędów, które mogą, ale nie muszą dawać błędnego wyniku asemblacji. Gdy nie wykryto żadnych błędów, wówczas drukowane jest "-- BŁĘDY (ÓW) ".

6.3. Sposób wywoływania i sterowania pracą programu CREF80

Program CREF80 stanowi pomicniczy program, stosowany w powiązaniu z programem makroasemblera M80. CREF80 służy do otrzymywania listingu z dodatkowymi danymi, zawierającymi informacje o odwołaniach wewnątrz asemblowanego modułu programu źródłowego. Postać takiego listingu opisano w pkt. 5.3. Chcąc wykorzystać program CREF80 należy przy ustalaniu warunków pracy makroasemblera użyć parametru /C /pkt. 6.1./. Użycie tego parametru powoduje wygenerowanie przez makroasembler zamiast normalnego listingu /zbiór typu PRN/, specjalnego zbioru z odwołaniami, oznaczonego typem CRF. Dzięki temu po zakończeniu tak przeprowadzonej asemblacji można wywołać dodatkowy program CREF80.

Wywołanie programu CREF dokonuje się przez wprowadzenie zlecenia nierezydentnego:

CREF80

*

W odpowiedzi na to program CREF80 zgłasza się przez wydrukowanie znaku gwiazdki. W celu uzyskania pliku zawierającego rozszerzony listing /oznaczony standardowo jako typ LST/ należy wprowadzić z konsoli /po gwiazdce/ linię dyrektywy

= nazwa programu

gdzie, nazwa programu oznacza nazwę pliku typu CRF,

który stanowi podstawę do uzyskania po przetwarzaniu, potrzebnego pliku z rozszerzonym listingiem.

Wywołanie programu CREF80 i podanie nazwy pliku typu CRF, może być też dokonane w jednym wierszu przez wprowadzenie z konsoli:

CREF80 = nazwa programu

Dla wyprowadzenia otrzymanego, rozszerzonego listingu bezpośrednio na drukarkę lub konsolę systemową, należy przed znakiem równości wypisać nazwę urządzenia według oznaczeń przyjętych w systemie operacyjnym CP/J.

Przykładowo, w celu wyprowadzenia na drukarkę rozszerzonego listingu, należy wprowadzić po zakończeniu asemblacji:

CREF80 LST: = nazwa programu

7. POSTAĆ PROGRAMU ŹRÓDŁOWEGO I DYREKTYWY

7.1. Format programów źródłowych

7.1.1. Uwagi ogólne

Program źródłowy powinien być przygotowany jako ciąg istrukcji pisanych w języku makroasemblera dla mikroprocesora INTEL 8080 lub Z80.

Zestawy instrukcji mnemonicznych dla obu tych mikroprocesorów /INTEL 8080 i Z80/ podano w Załączniku B.

Szczegóły dotyczące opisu listy rozkazów mikroprocesorów są zawarte w dostępnej dokumentacji dotyczącej tych mikroprocesorów.
Dyrektywy /pseudoinstrukcje/ dla makroasemblera M80 zostały opisane w dalszej części podręcznika a zestawienie dyrektyw podano w Załączniku A.

Długość linii programu źródłowego nie może przekraczać 132 znaków. M80 przyjmuje i zachowuje znaki małych liter dla łańcuchów znaków i komentarzy, a dla symboli oznaczeń kodu operacji i dyrektyw pisanych z wykorzystaniem małych liter przeprowadza zmianę tych liter na duże.

7.1.2. Instrukcje w języku makroasemblera

Instrukcja programu źródłowego ma postać:

etykieta : operator operandy komentarz

Instrukcja nie musi zaczynać się w pierwszej kolumnie. Dla zwiększenia czytelności programu można stosować odstępy w postaci wielu spacji lub tabulację poziomą.

Jeśli linia programowa posiada etykietę, wówczas etykieta występuje jako pierwsza w linii, a bezpośrednio po niej występuje dwukropek. W przypadku, gdy po etykiecie występują dwa dwukropki, wówczas jest to równoważne deklaracji tej etykiety jako GLOBAL /patrz pkt. 7.3.2.7./.

Przykładowo, zapis

```
COS::  RET
```

jest równoważny zapisowi

```
COS:    GLOBAL  COS
        RET
```

Następną pozycją instrukcji po etykiecie /lub pierwszą jeśli nie występuje etykieta/ jest operator.

Operatorem może być oznaczenie mnemoniczne kodu operacji mikroprocesora INTEL 8080 lub Z80, dyrektywa /pseudoinstrukcja/, makrowywołanie lub wyrażenie. Kolejność oceny operatora jest następująca:

1. makrowywołanie,
2. mnemonik kodu operacji/dyrektywa,
3. wyrażenie.

Makroasembler w przypadku napotkania wyrażenia nie traktuje tego jako błąd, lecz tak, jakby to była instrukcja z dyrektywą DB /pkt. 7.3.2.1./.
Operandy występujące po operatorze mają różną postać, odpowiednio do tego operatora.
Komentarz zawsze jest rozpoczynany znakiem średnika i zakończony znakiem CR.
Komentarz może występować w wierszu samodzielnie, lub też jako uzupełnienie do zdania występującego w danym wierszu.
Rozszerzone komentarze mogą być wprowadzone dyrektywą COMMENT /pkt. 7.3.4.1./.

7.1.3. Nazwy

Nazwy dla makroasemblera M80 mogą być dowolnej długości, jednak tylko cześć pierwszych znaków jest traktowanych jako znaczące.

Dopuszczalne jest stosowanie następujących znaków /w ramach przedziałów/

A - Z   0 - 9 $,   ?   o

Dopuszczalne jest również stosowanie w nazwie znaku kreski podkreślającej -. Symbol nie może zaczynać się cyfrą. Przy czytaniu symboli małe litery są zamieniane na duże.
Jeśli odwołanie do symbolu zakończone jest znakami, wówczas jest to równoważne deklarowaniu tego symbolu jako zewnętrznego /patrz dyrektywa EXT, pkt.7.3.2.6/.

7.1.4. Stałe numeryczne

Stałe numeryczne, jeśli nie jest to inaczej ustalone, są traktowane jako liczby dziesiętne. Zmiany (konwersji) można dokonać przy pomocy dyrektywy RADIX /pkt. 7.3.4.5./. Można ustalać dowolną podstawę liczbową z zakresu od 2 /binarną/ do 16 /heksadecymalną/.
Dla podstawy liczbowej większej od 10 stosuje się oznaczenia cyfr powyżej 9 przy pomocy liter A, B, C, D, E i F. Jeśli pierwsza cyfra w liczbie jest inna niż 0 - 9, wówczas zapis tej liczby powinien być poprzedzony cyfrą 0.
Liczby są 16-to bitowymi wartościami bez znaku.
Liczba jest traktowana jako zapis dla bieżącej, obowiązującej wartości podstawy, jeśli nie zaznaczono inaczej, w specjalny, niżej podany sposób:

n n n n B	dla liczby binarnej
n n n n D	dla liczby dziesiętnej
n n n n O	dla liczby oktalnej
n n n n Q	
n n n n H	dla liczby heksadecymalnej

X'n n n n'

W przypadku przekroczenia przez liczbę wartości ponad 16 bitów, nadmiar jest ignorowany, a jako wynik traktuje się 16 najmniej znaczących bitów.

7.1.5. Stałe znakowe

Stała znakowa jest ciągiem złożonym z zerowej liczby znaków, jednego lub dwóch znaków umieszczonych między znakami cudzysłowów / pojedyńczych lub podwójnych/ i występującym w złożonym wyrażeniu.

Dla przykładu w zdaniu

```
DB  A + 1
```

A stanowi stałą znakową, lecz w zdaniu

```
DB  A
```

A stanowi łańcuch znaków, ponieważ występuje w prostym wyrażeniu.

Zasady stosowania ograniczników /delimiterów/ dla stałych znakowych i łańcuchów znaków są takie same.

Stała znakowa zawierająca jeden znak posiada wartość odpowiadającą wartości kodu ASCII tego znaku. Oznacza to, że starszy bajt 16-to bitowej reprezentacji wartości jest równy zero, a młodszy bajt tej reprezentacji jest równy wartości kodu ASCII tego znaku.

Przykładowo, wartością stałej A jest 41H.

Stała znakowa zawierająca dwa znaki posiada wartość starszego bajtu, równą wartości kodu ASCII pierwszego

znaku, a wartość młodszego bajtu równą wartości kodu ASCII drugiego znaku.

Dla przykładu, wartość stałej znakowej AB jest równa 41H 256+42H.

7.1.6. Łańcuchy znaków

Łańcuch znaków stanowi ciąg zerowej lub większej liczby znaków umieszczony między znakami cudzysłowu /pojedyńczych lub podwójnych/. Cudzysłowy stanowią tu ograniczniki. Dla przedstawienia jednego znaku cudzysłowu wewnątrz łańcucha muszą być użyte dwa kolejne znaki takiego cudzysłowu.

Przykładowo zdanie:

```
DB   "TO JEST"  "COS"  "DLA MNIE"
```

powoduje zapamiętanie łańcucha:

```
TO JEST "COS" DLA MNIE
```

7.2. Wyznaczanie wartości wyrażeń

7.2.1. Operatory arytmetyczne i logiczne

Dopuszczalne jest stosowanie niżej podanych operatorów, wymienionych w kolejności priorytetu od najwyższego do najniższego:

NUL
LOW, HIGH
x,/, MOD, SHR, SHL
-
+, -
EQ, NE, LT, LE, GT, GE

NOT

AND

OR, XOR

Dla zmiany kolejności działania operatorów można używać nawiasów. Przy wyznaczaniu wartości wyrażeń, fragmenty wyrażeń zawierające operatory o wyższym priorytecie są obliczone w pierwszej kolejności. Wszystkie operatory z wyjątkiem: +, -, x, /powinny być oddzielone od ich operandów co najmniej jedną spacją.

Znaczenie poszczególnych operatorów jest następujące:

NUL - daje wartość logiczną zależną od tego, co znajduje się na prawo od tego operatora do końca wiersza.
Dyrektywa warunkowa /pkt. 7.3.7./.
IF NUL argument
jest nieprawdą, jeśli pierwszym znakiem argumentu jest inny znak niż średnik lub znak powrotu karetki CR.

TYPE - daje bajt opisujący właściwości argumentu tego operatora.
Argumentem może być dowolne wyrażenie.
Gdy wyrażenie jest nieprawidłowe, otrzymywany bajt ma wartość zero. Dwa najmłodsze bity otrzymanego bajtu określają odpowiednio tryb wyrażenia:

0 - tryb absolutny,
1 - z przesuwalnym programem,
2 - z przesuwalnymi danymi,
3 - COMMON.

Najwyższy bit bajtu /80H/ jest równy 1, jeśli wyrażenie jest zewnętrzne, a równy 0 gdy wyrażenie jest lokalne.
Bit szósty /20H/ jest równy 1 tylko wtedy, gdy wyrażenie jest lokalnie definiowane.

LOW - wydziela 8 najniższych bitów z 16-to bitowej wartości wyrażenia.

HIGH - wydziela 8 najwyższych bitów z 16-to bitowej wartości wyrażenia.

/ - dzielenie

MOD - moduls. Dzieli lewy operand przez prawy operand i zwraca wartość stanowiącą resztę z tego dzielenia.

SHR - przesunięcie w prawo. Po SHR jest podawana liczba całkowita, która określa liczbę przesunięć bitowych pozycji w prawo.

SHL - przesunięcie w lewo. Po SHL jest podawana liczba całkowita, która określa liczbę przesunięć bitowych pozycji w lewo.

\- - minus unarny. Wskazuje, że zapisana z prawej strony wartość jest ujemna.

\+ - dodawanie

\- - odejmowanie

Operatory dające wartości logiczne 1 /TRUE/ lub 0 /FALSE/

EQ - równy. Daje 1, gdy operandy są równe.

NE - różny. Daje 1, gdy operandy są różne.

LT - mniejszy niż. Daje 1, gdy lewy operand jest mniejszy od prawego.

LE - mniejszy lub równy. Daje 1, gdy operand z lewej stronv jest mniejszy niż prawy operand lub równy.

GT - większy. Daje 1, gdy lewy operand jest większy od prawego.

GE - większy lub równy. Daje 1, gdy lewy operand jest większy niż prawy operand lub równy.

NOT - operacja logiczna. Daje 1, gdy lewy operand jest równy 1 a prawy 0 lub gdy lewy operand jest równy 0 a prawy 1.
Jeżeli oba operandy mają wartość 0 lub wartość 1, to uzyskuje się wynik 0.

AND - mnożenie logiczne. Daje 1, gdy oba operandy są równe 1. Obydwa operandy muszą być w trybie absolutnym.

OR - suma logiczna. Daje 1, gdy jeden z operandów lub obydwa mają wartość 1. Obydwa operandy muszą być w trybie obsolutnym.

7.2.2. Tryby wyrażeń

Symbole występujące w wyrażeniach jako operandy mogą być w jednym z poniższych trybów:

- w trybie absolutnym,
- z przesuwalnymi danymi,
- z przesuwalnym programem /kodem/,
- w trybie COMMON.
  /patrz pkt. 7.3.3. opisujący dyrektywy ASEG, CSEG i COMMON.

Symbole asemblowane przy działaniu dyrektywy ASEG są w trybie absolutnym, przy działaniu dyrektywy CSEG /co jest standardem, jeśli nie określono inną dyrektywą inaczej/ są w trybie z przesuwalnym programem, a przy działaniu dyrektywy DSEG są w trybie z przesuwalnymi danymi.

Występowanie trybów COMMON w danym programie jest określone przez ilość bloków COMMON, które zostały nazwane przy pomocy dyrektywy COMMON. Dwa symbole COMMON nie są w tym samym trybie, jeśli nie należą do tego samego bloku COMMON.

Dla dowolnych operacji innych niż dodawanie lub odejmowanie operandy muszą być podane w trybie absolutnym.

Dla operacji dodawania stosuje się następujące reguły:

a/ co najmniej jeden z dwóch operandów powinien być w trybie absolutnym,

b/ tryb absolutny + tryb x = tryb x.

Dla operacji odejmowania obowiązują następujące reguły:

a/ tryb x - tryb absolutny = tryb x,

b/ tryb x - tryb x = tryb absolutny /jeśli tryby po lewej stronie są takie same/.

Każdy pośredni etap oceny wyrażenia musi spełniać powyższe zasady dotyczące trybów, gdyż w przeciwnym przypadku jest sygnalizowany błąd.

Przykładowo, jeśli PIK, LIS, ZEZ są symbolami w trybie przesuwalnym, wówczas wyrażenie

PIK = LIS - ZEZ

będzie sygnalizowane błędem R, ponieważ pierwszy krok /PIK + LIS/ powoduje dodawanie dwu relokowalnych wartości /jedna powinna być w trybie absolutnym/.

Trudność tę można pokonać przez dodanie nawiasów

PIK = /LIS - ZEZ/

Powyższe wyrażenie jest akceptowane, ponieważ /LIS - ZEZ/ powoduje wygenerowanie wartości w trybie absolutnym, która w następnym kroku jest dodawana do wartości PIK w trybie z przesuwalnym programem.

7.2.3. Symbole zewnętrzne

Niezależnie od trybu, w którym występuje, symbol może być określony jako zewnętrzny /external/ lub nie /patrz dyrektywa EXT/BYTE EXT, pkt. 7.3.2.6/. Pojęcie symbolu zewnętrznego oznacza, że jest on zdefiniowany na zewnątrz modułu programu, w którym jest używany.
W module tym, gdzie jest definiowany, symbol taki powinien być zadeklarowany jako GLOBAL /PUBLIC/.

Dla symboli zewnętrznych obowiązują następujące zasady stosowania ich w wyrażeniach:

- symbole zewnętrzne mogą być w wyrażeniach zawierających tylko operatory: +, -, x, /, MOD, HIGH, LOW,
- wyrażenie, w którym występują symbole zewnętrzne ma wartość, która też jest typu zewnętrznego.

7.2.4. Kody operacji jako operandy

Kody operacji mikroprocesora INTEL 8080 mogą być używane jako jednobajtowe operandy.

Oto przykłady poprawnych zapisów:

```
MVI  A /JMP/
CPI    /INX H/
MVI  C, /MOV A,B/
ADI    /CPI/
ACI    /LXI B/
```

Tylko pierwszy bajt w instrukcji może być wykorzystywany przy zapisie, gdyż w przeciwnym przypadku jest sygnalizowany błąd.

Np. niepoprawne jest używanie jako operandów: /CPI 5/, /LXI B, COSA1/lub /JMP COS2/.

Kody operacji używane jako operandy nie muszą być zapisywane w nawiasach.

UWAGA: Przy pracy makroasemblera w trybie Z80 nie można używać kodów operacji jako operandów.

7.3. Dyrektywy w programie źródłowym

7.3.1. Dyrektywy wyboru zestawu instrukcji

7.3.1.1..Z80 - instrukcje mikroprocesora Z80

Dyrektywa .Z80 jest bez argumentów i umożliwia przyjmowanie przez makroasembler M80 instrukcji odpowiadających mikroprocesorowi Z80 /patrz Załącznik B/. Makroasembler może pracować też w trybie .Z80 przez dodanie parametru /Z przy wywołaniu makroasemblera /pkt. 6.1/.

7.3.1.2. .8080 - instrukcje mikroprocesora 8080

Dyrektywa .8080 jest bez argumentów i umożliwia przyjmowanie przez makroasembler M80 instrukcji odpowiadających mikroprocesorowi INTEL 8080 /patrz Załącznik B/.
Jest to standardowy sposób pracy makroasemblera M80, gdy brak jest dyrektywy wyboru zestawu instrukcji.

7.3.2. Dyrektywy definiowania danych i symboli

Wszystkie podane niżej dyrektywy definiowania danych i symboli mogę być stosowane zarówno w programach dla mikroprocesora 8080, jak i Z80, z wyjątkiem dyrektywy SET, która jest niedopuszczalna dla Z80.
Gwiazdkami z lewej strony oznaczono dyrektywy zwykle stosowane dla Z80. Brak gwiazdki oznacza dyrektywę stosowaną dla 8080.

7.3.2.1. DB - definiowanie bajtów danych

Postać:

DB wyrażenie, wyrażenie ...

DEFB wyrażenie wyrażenie ...

DB łańcuch łańcuch ...

DEFM łańcuch łańcuch ...

Argumentem dla dyrektywy może być wyrażenie lub łańcuch.
Dyrektywa powoduje zapamiętanie wartości wyrażeń lub łańcuchów znaków w kolejnych komórkach pamięci, poczynając od komórki wyznaczonej przez bieżącą wartość licznika komórek.

Wyrażenie powinno odpowiadać wartości jednego bajtu /jeśli przekracza 255, sygnalizowany jest błąd A/.

Znaki występujące w łańcuchach są zapamiętywane w kolejności występowania, przy czym każdemu znakowi odpowiada jeden bajt z najbardziej znaczącym bitem równym 0.

Przykład asemblacji:

0000'	41	42		DB	AB		
0002'	42			DB	AB	AND	OFFH
0003'	41	42	43	DB	ABC		

7.3.2.2. DC - definiowanie znaków

Postać:

DC łańcuch

Dyrektywa DC powoduje zapamiętanie znaków łańcucha w kolejnych komórkach pamięci, poczynając od bieżącej wartości licznika komórek.
Podobnie jak dla dyrektywy DB znaki są zapamiętywane w kolejności występowania, przy czym znakowi odpowiada jeden bajt z najbardziej znaczącym bitem równym zero. Jednak w odróżnieniu od dyrektywy DB, dyrektywa DC powoduje zapamiętanie ostatniego znaku łańcucha w postaci bajtu z najbardziej znaczącym bitem równym jeden. W przypadku, gdy argumentem dyrektywy DC jest pusty łańcuch, wówczas sygnalizowany jest błąd.

7.3.2.3. DS - definiowanie bajtów pamięci

Postać:

DS wyrażenie, wartość

DEFS wyrażenie wartość

Dyrektywa DS rezerwuje obszar pamięci o ilości bajtów wyznaczonej wartością wyrażenia.

Wartość tego wyrażenia przy pierwszym przebiegu asemblacji powinna być możliwa do określenia, gdyż w przeciwnym przypadku sygnalizowany jest błąd.

Jeżeli wymagane jest, aby zarezerwowany obszar pamięci miał komórki o ustalonej zawartości, wówczas należy określić również wartość odpowiadającą temu, co jest umieszczone we wszystkich komórkach.
Jeżeli nie podano wartości w dyrektywie, wówczas obszar zarezerwowanych komórek nie ulega zmianie /nie jest inicjowany przy ładowaniu programu wynikowego/.
Rezerwowany obszar komórek może być zerowany przez podanie wartości równej zero lub użycie parametru /M przy wywołaniu makroasemblera /pkt. 6.1/.

7.3.2.4. DW - definiowanie słów

Postać:

DW   wyrażenie, wyrażenie ...

DEFW wyrażenie wyrażenie ...

Dyrektywa DW powoduje zapamiętanie wartości wyrażeń w kolejnych komórkach pamięci, poczynając od komórki wyznaczonej przez bieżącą wartość licznika komórek. Wartości wyrażeń są określone jako zawartości słów dwubajtowych.

7.3.2.5. EQU - przyrównanie

Postać:

nazwa EQU wyrażenia

Dyrektywa EQU przypisuje wartość określoną przez wyrażenie symbolowi określonemu przez nazwę.
Jeśli wyrażenie jest typu zewnętrznego, wówczas jest sygnalizowany błąd. Jeżeli symbol posiada już wartość inną, niż określa to wyrażenie, wówczas sygnalizowany jest błąd M.

7.3.2.6. EXT/BYTE EXT - zewnętrzny symbol

Postać:

```
EXT      nazwa ,  nazwa ...
EXTRN    nazwa ,  nazwa ...
EXTERNAL nazwa ,  nazwa ...

BYTE  EXT      symbol
BYTE  EXTRN    symbol
BYTE  EXTERNAL symbol
```

Dyrektywy powyższe stanowią deklaracje symbolu /symboli/, który jest zewnętrzny, tzn. jest zdefiniowany w innym module programu.
Jeśli symbol ten jest zdefiniowany w bieżącym module programu, wówczas sygnalizowany jest błąd M.

Zewnętrzne symbole mogą być określone przez dwa lub jeden bajt /nazwy dyrektyw z "BYTE"/.
Dla wszystkich zewnętrznych symboli tylko 6 pierwszych znaków jest brane pod uwagę a reszta jest pomijana.

Odwołanie do nazwy, po której bezpośrednio zapisano dwa znaki // // /np. NAZWA // // /, jest równoważne deklarowaniu tej nazwy jako zewnętrznej.

7.3.2.7. GLOBAL - globalny symbol

Postać:

```
ENTRY   nazwa , nazwa ...
GLOBAL  nazwa , nazwa ...
PUBLIC  nazwa , nazwa ...
```

Dyrektywy powyższe stanowią deklaracje nazw, które są traktowane jako globalne, to znaczy są dostępne nie tylko w danym module programu, w którym występują, ale są również udostępnione innym modułom programów /w trakcie działania programu łącząco-ładującego L80/.
Przy rozpatrywaniu nazwy globalnej tylko 6 pierwszych znaków jest branych pod uwagę a reszta jest pomijana.

7.3.2.8. SET - ustawianie

Postać:

```
nazwa  SET  wyrażenie /nie używane dla trybu pracy
                        .Z80/
nazwa  DEFL wyrażenie
nazwa  ASET wyrażenie
```

Dyrektywy powyższe przypisują wartość określoną przez wyrażenie danej nazwie.
Nazwa może być etykietą, symbolem lub zmienną i może występować w wyrażeniach.
Po nazwie nie powinien występować dwukropek.
Gdy wyrażenie jest zewnętrzne, sygnalizowany jest błąd.

Dyrektywy SET nie wolno stosować w trybie pracy makroasemblera .Z80, gdyż jest ona mnemonikiem instrukcji dla Z80.
Można w tym przypadku używać dyrektyw DEFL i ASET.
Działanie dyrektywy SET jest podobne do dyrektywy EQU, z tym tylko wyjątkiem, że określona nazwa dla SET może mieć nadawane różne wartości przez wielokrotne użycie dyrektywy SET /lub DEFL, ASET/, a dyrektywą EQU można nadawać wartość danej nazwie tylko raz.

7.3.3. Dyrektywy trybu asemblacji

W czasie asemblacji wartości symboli są określone odpowiednio w odniesieniu do licznika komórek /licznika rozkazów/ pamięci, w której mają być umieszczone.
Tryb pracy asemblera w momencie analizy danej instrukcji programu źródłowego decyduje o miejscu pamięci, w którym ma być umieszczony otrzymany kod wynikowy.
Kod wynikowy może być umieszczony w ściśle określonych przez programistę komórkach pamięci i wówczas jest to tzw. kod absolutny. Obok tego występuje też tzw. kod przesuwalny /relokowalny/ odpowiadający temu, że dopiero na jego podstawie w procesie otrzymywania ostatecznego kodu wynikowego następuje bezwzględne przyporządkowanie określonym fragmentom programu źródłowego, określonych miejsc w pamięci.
Oprócz powyższego, istnieje możliwość przyporządkowywania otrzymanego kodu przesuwalnego do wybranych obszarów pamięci, zwanych segmentami i odpowiadających docelowo rozdzieleniu pamięci operacyjnej na pamięć typu RAM /z możliwością odczytu i zapisu/ oraz ROM/PROM /tylko odczytywalną, bez możliwości zapisu/.

Rozróżniamy segment odpowiadający asemblacji w trybie przesuwalnego programu /instrukcji/, intencjonalnie związany z pamięcią ROM oraz segment odpowiadający asemblacji w trybie przesuwalnych danych, intencjonalnie związany z pamięcią RAM.

Niezależnie od tego istnieje potrzeba, aby różne moduły programu asemblowane oddzielnie, posiadały przy współpracy wspólne, jednoznacznie określone obszary pamięci. Uzyskuje się to dzięki wprowadzeniu asemblacji wybranych odcinków programu w trybie zwanym COMMON. Stąd w czasie asemblacji mogą wystąpić 4 tryby asemblacji:

- tryb absolutny,
- tryb przesuwalnego programu,
- tryb przesuwalnych danych,
- tryb COMMON.

Każdy z powyższych trybów asemblacji jest wyznaczony przez wystąpienie odpowiedniej dyrektywy, ustalającej ten tryb.
Określony tryb asemblacji obowiązuje aż do wystąpienia następnej dyrektywy zmieniającej ten tryb asemblacji.

Gdy rozpoczęła się asemblacja i jeśli nie została użyta dyrektywa ustalająca tryb asemblacji, to obowiązuje tryb przesuwalnego programu. Jest to warte podkreślenia, gdyż dla innych makroasemblerów, dających przesuwalny kod jako stan pierwotny występuje zwykle asemblacja w trybie absolutnym.

7.3.3.1. ASEG - absolutny tryb asemblacji

Dyrektywa ASEG powoduje rozpoczęcie asemblacji w trybie absolutnym. Jeśli nie jest ustalone inaczej, pierwsza komórka pamięci, w której jest umieszczony kod wynikowy, ma adres 0.
Ponieważ najniższe adresy w pamięci są zarezerwowane dla systemu operacyjnego, zaleca się, aby po dyrektywie ASEG stosować dyrektywę ORG, ustalającą adres początkowy o wartości 103H lub większej.

Dyrektywa ASEG nie ma żadnych parametrów.

7.3.3.2. CSEG - asemblacja w trybie przesuwalnego programu

Dyrektywa CSEG powoduje rozpoczęcie asemblacji w trybie przesuwalnego programu.
Jeśli nie jest użyta dyrektywa ORG, translacja przebiega tak, jakby pierwsza komórka pamięci /w segmencie przesuwalnego programu/, miała mieć adres względny 0.
Zmianę tego adresu można dokonać przez użycie dyrektywy ORG po dyrektywie CSEG.
Należy jednak pamiętać, że dyrektywa ORG w tym przypadku nie ustala absolutnego adresu w segmencie przesuwalnym programu.
Dyrektywa ORG powoduje natomiast dodanie do bieżącej wartości licznika komórek w segmencie przesuwalnego programu, wartości będącej parametrem tej dyrektywy.

Przy rozpoczynaniu asemblacji, założony stan początkowy jest taki, jak by była użyta dyrektywa CSEG.

Dyrektywa CSEG nie ma żadnych parametrów.

7.3.3.3. DSEG - asemblacja w trybie przesuwalnych danych

Dyrektywa DSEG powoduje rozpoczęcie asemblacji w trybie przesuwalnych danych.
Jeśli nie jest ustalone inaczej, adres pierwszej komórki pamięci /w segmencie przesuwalnych danych/, w której umieszczony jest przesuwalny kod wynikowy, wynosi 0.
Zmiany tego adresu można dokonać przez użycie dyrektywy ORG po dyrektywie DSEG.
Należy jednak pamiętać, że dyrektywa ORG w tym przypadku nie ustala absolutnego adresu w segmencie przesuwalnych danych.
Dyrektywa ORG powoduje natomiast dodanie do bieżącej wartości licznika komórek w segmencie przesuwalnych danych wartości, będącej parametrem tej dyrektywy.
Dyrektywa DSEG nie ma żadnych parametrów.

7.3.3.4. COMMON - asemblacja w trybie COMMON

Postać:

COMMON /nazwa bloku/

Dyrektywa COMMON tworzy wydzielony blok w pamięci, stanowiący wspólny obszar do współpracy różnych modułów programowych.
Bloki w pamięci tworzone są oddzielnie dla każdego COMMON o określonej nazwie.
Jeśli w dyrektywie nie występuje nazwa bloku lub zamiast nazwy występują spacje, to blok COMMON stanowi COMMON bez nazwy.

Zdania, które są asemblowane w trybie COMMON odpowiadają zajmowaniu wydzielonego obszaru pamięci /dane, tablice, zmienne/ i nie stanowią instrukcji wykonywanych przez mikroprocesor.

Po wystąpieniu dyrektywy COMMON rozpoczyna się asemblacja kolejnych zdań programu źródłowego z umieszczeniem wyniku asemblacji w obszarze pamięci, przeznaczonym dla bloku COMMON o określonej nazwie. Asemblacja do bloku COMMON kończy się po wystąpieniu dyrektywy zmieniającej tryb asemblacji.
Długość bloku COMMON jest uzależniona od asemblowanych zdań.
Bloki COMMON o tej samej nazwie, asemblowane dla różnych modułów programu mogą mieć różne długości. W tym przypadku, w czasie działania programu łącząco-ładującego L80 moduł z najdłuższym blokiem COMMON powinien być ładowany w pierwszej kolejności.

Omawiane bloki COMMON mogą też służyć do współpracy z programami pisanymi w języku FORTRAN.

7.3.3.5. ORG - ustawianie licznika komórek

Postać:

ORG wyrażenie

Dyrektywa ORG służy do zmiany zawartości licznika komórek.
Przy asemblacji w trybie absolutnym licznik komórek jest ustawiony na wartość odpowiadającą wartości wyrażenia, będącego parametrem dyrektywy ORG.

Przy asemblacji w trybie przesuwalnego programu /CSEG/, przesuwalnych danych /DSEG/ i w trybie COMMON, licznik komórek odpowiadający danemu trybowi asemblacji jest zwiększany o wartość określoną przez parametr dyrektywy ORG.

Wszytkie nazwy występujące w wyrażeniu powinny mieć wartości możliwe do określenia w pierwszym przebiegu makroasemblera i powinny to być wartości absolutne lub tego samego rodzaju co tryb asemblacji.

7.3.3.6. .PHASE/.DEPHASE - relokowanie

Postać:

```
.PHASE    wyrażenie
   .
   .
   .
.DEPHASE
```

Dyrektywa .PHASE pozwala na umieszczenie kodu wynikowego w jednym obszarze, a wykonanie go w innym obszarze począwszy od adresu określonego przez wartość wyrażenia. Powinno to być wyrażenie absolutne. Stowarzyszona dyrektywa .DEPHASE służy do wskazania końca bloku programu, którego kod ma być relokowany.
Tryb asemblacji dla zdań wewnątrz relokowanego odcinka programu jest absolutny.
Wydzielenie bloku .PHASE... .DEPHASE umożliwia wykonanie wybranego bloku programu dla wybranych adresów, zdefiniowanych jako absolutne.

7.3.4. Dyrektywy dotyczące pliku z programem źródłowym

7.3.4.1. .COMMENT - długi komentarz

Postać:

.COMMENT ogranicznik tekst ogranicznik

Dyrektywa .COMMENT umożliwia wprowadzenie do tekstu źródłowego programu długich komentarzy, obejmujących też wiele wierszy. Uwalnia ona od obowiązku rozpoczynania wiersza komentarza przez znak średnika. Pierwszy niepusty znak występujący za nazwą dyrektywy jest traktowany jako ogranicznik. Tekst za tym ogranicznikiem traktowany jest jako komentarz aż do wystąpienia powtórnego znaku ogranicznika.

7.3.4.2. END - koniec programu

Postać

END wyrażenie

Dyrektywa END określa koniec programu źródłowego. Jeżeli dyrektywa ta nie występuje, wówczas wypisywana jest informacja o błędzie na końcu listingu /jako ostrzeżenie/. Wyrażenie jeżeli występuje określa adres, który program łącząco-ładujący przyjmuje jako adres startu programu. Wówczas pod adresem 100H umieszczona jest instrukcja skoku mikroprocesora 8080 /JMP/ pod ten adres.

7.3.4.3. INCLUDE/MACLIB - dołącz plik

Postać:

```
INCLUDE    nazwa pliku
$INCLUDE   nazwa pliku
MACLIB     nazwa pliku
```

Powyższe dyrektywy dołączają w określone miejsce programu źródłowego fragmenty programu w postaci źródłowej z pliku o określonej nazwie w argumencie dyrektywy.
Z założenia program dołączany jest w pliku typu MAC. Jeśli typ pliku, z którego jest dołączany fragment programu jest inny, wówczas należy podać ten typ w dyrektywie.
Nie wolno zagnieżdżać dyrektyw do łączenia pliku. Użycie takiej operacji sygnalizuje błąd O.
Jeśli dołączany plik nie istnieje, wówczas jest sygnalizowany błąd V.

Na listingu dołączony fragment programu źródłowego jest oznaczony literą C, drukowaną w każdym wierszu między uzyskanym kodem wynikowym i zawartością wiersza programu źródłowego.

7.3.4.4. NAME - nazwa modułu

Postać:

NAME / nazwa modułu

Dyrektywa NAME nadaje nazwę modułowi programu. Wypisanie nawiasów i znaków cudzysłowu jest konieczne.

Jako nazwę modułu używa się tylko 6 pierwszych znaków podanej nazwy.
Moduł może mieć określoną nazwę również dyrektywą TITLE.
Jeśli nie użyto żadnej z tych dyrektyw, wówczas moduł otrzymuje nazwę odpowiadającą nazwie pliku z programem źródłowym.

7.3.4.5. .RADIX - podstawa liczb

Postać:

.RADIX wyrażenie

Dyrektywa .RADIX pozwala ustalać podstawę liczb występujących w programie źródłowym, w zakresie od 2 do 16. Standardowo, jeżeli nie określono inaczej, wszystkie stałe są przedstawiane w postaci dziesiętnej.
Wartość wyrażenia będącego argumentem tej dyrektywy jest traktowana zawsze jako dziesiętna reprezentacja, niezależna od bieżącej podstawy liczbowej w programie.

Dyrektywa .RADIX nie wpływa na postać listingu.
Przy ustalonej tą dyrektywą podstawie liczb nie trzeba specjalnie oznaczać zapisanych liczb o tej podstawie.
Liczby o innej podstawie trzeba wyróżniać specjalnymi oznaczeniemi /pkt. 7.1.4/.

7.3.4.6. .REQUEST - moduły związane

Postać:

.REQUEST nazwa pliku nazwa pliku ...

Dyrektywa ta jest przeznaczona do wykorzystania podczas działania programu łącząco-ładującego L80. Daje ona informację dla programu łącząco-ładującego o tym, że w przypadku wystąpienia zewnętrznych symboli, które są niezdefiniowane w zasemblowanym module programu, należy dla określenia wartości tych symboli wykorzystać dodatkowe moduły podane tą dyrektywą.

Nazwy plików nie wymagają określenia typów tych plików, a które powinny być z założenia typu REL.

7.3.5. Dyrektywy sterujące listingiem

7.3.5.1. PAGE/ x EJECT - nowa strona

Postać:

PAGE	wyrażenie	
EJECT	wyrażenie	/dyrektywa dla trybu pracy .Z80, jedna gwiazdka stanowi część dyrektywy/
* EJECT		

Dyrektywy te powodują wyprowadzenie nowej strony listingu.
Znak nowej strony w listingu jest umieszczony przy końcu strony. Jeśli jest podana wartość wyrażenia, to określa ona wielkość nowej strony /w wierszach na stronę/ i powinna być z zakresu od 10 do 255.

Gdy nie podano wielkości nowej strony, to wynosi ona 50 wierszy na stronę.
Makroasembler powoduje przejście do drukowania nowej strony po zadrukowaniu 58 wierszy.

7.3.5.2. TITLE - tytuł

Postać:

TITLE tekst

Dyrektywa TITLE określa tytuł, który jest drukowany w pierwszym wierszu każdej strony listingu.
Jeżeli nie użyto dyrektywy NAME, wówczas 6 pierwszych znaków tytułu jest wykorzystywane jako nazwa modułu.
Gdy nie użyto dyrektywy TITLE ani dyrektywy NAME, to nazwa modułu jest utworzona z nazwy pliku programu źródłowego.

Dyrektywa TITLE powinna być użyta tylko raz.
W przypadku kilkakrotnego jej użycia sygnalizowany jest błąd Q.

7.3.5.3. SUBTTL / $ TITLE - podtytuł

Postać:

SUBTTL tekst

O TITLE tekst

Dyrektywy te określają podtytuł, który jest drukowany na każdej stronie listingu, w drugim wierszu.
Tekst podany jako argument jest obcinany po pierwszych 60 znakach. W programie może wystąpić dowolna liczba dyrektyw tego typu.

Po napotkaniu nowej dyrektywy SUBTTL, makroasembler zamienia dotychczasowy podtytuł na nowo otrzymany.

Aby zaprzestać drukowania podtytułu należy użyć dyrektywę SUBTTL z zerową długością łańcucha dla tekstu /tz. bez argumentu/.

7.3.5.4. .LIST i .XLIST - włączenie i wyłączenie listingu

Postać:

.LIST - włączanie listingu

.XLIST - wyłączanie listingu

Jako sytuację standardową przyjmuje się, że listing jest włączony.
Jeśli w dyrektywie sterującej pracą makroasemblera /pkt. 6.1/ wyspecyfikowano plik listingu, to będzie on tworzony.
Gdy w programie źródłowym wystąpi .XLIST, wówczas listing nie jest wyprowadzany.

Działanie .XLIST pozostaje w mocy do wystąpienia dyrektywy .LIST.

Jeżeli wystąpi .XLIST, działania innych dyrektyw dotyczących listingu, z wyjątkiem LIST nie powodują wyprowadzenia listingu.

7.3.5.5. .PRINTX - wydruk

Postać:

.PRINTX ogranicznik tekst ogranicznik

Dyrektywa ta umożliwia wyprowadzenie na konsolę systemową tekstu występującego jako argument dyrektywy.
Pierwszy niepusty znak występujący za nazwą dyrektywy jest traktowany jako ogranicznik. Dalej występujący tekst jest drukowany aż do napotkania powtórnego znaku ogranicznika.

.PRINTX jest użyteczna dla monitorowania przebiegu pracy przy długiej asemblacji lub dla wyprowadzenia wartości argumentów dla warunkowej asemblacji.
Dyrektywa powoduje wydruk tekstów dla obu przebiegów asemblacji.
Jeżeli potrzebny jest wydruk tylko dla jednego z przebiegów, należy użyć dyrektyw warunkowej asemblacji IF1 lub IF2, w zależności od tego, dla którego przebiegu powinien być wyprowadzony tekst.

7.3.5.6. .SFCOND - wstrzymanie listingu nieasemblowanych fragmentów

Postać:

.SFCOND

Dyrektywa ta powoduje wstrzymanie wyprowadzenia na listing fragmentów programu niepodlegających asemblacji /w wyniku asemblacji warunkowej/.

7.3.5.7. .LFCOND - listing nieasemblowanych fragmentów

Postać:

.LFCOND

Dyrektywa ta powoduje podjęcie wyprowadzenia na listing fragmentów programu niepodlegających asemblacji /w wyniku asemblacji warunkowej/.

7.3.5.8. .TFCOND - przełączenie listowania nieasemblowanych fragmentów

Postać:

.TFCOND

Dyrektywa ta powoduje przełączenie stanu listowanie/nielistowanie nieasemblowanych fragmentów związanych z asemblacją warunkową. Stan początkowy jest ustalony przez parametr /X, w dyrektywie sterującej pracą makroasemblera /pkt. 6.1/.
Jeżeli parametr ten występuje, wówczas użycie .TFCOND powoduje listowanie nieasemblowanych fragmetnów.
Jeżeli parametr /X nie występuje, użycie .TFCOND wstrzymuje wyprowadzenie listingu nieasemblowanych fragmentów programu.

Dyrektywa .TFCOND działa niezależnie od dyrektyw .LFCOND i .SFCOND.

7.3.5.9. .XALL - listowanie wszystkich fragmentów poza nieasemblowanymi

Postać:

.XALL

Dyrektywa ta powoduje wyprowadzenie na listing rozwinięć makroinstrukcji i powtarzanych bloków, ale bez wyprowadzania fragmentów warunkowo nieasemblowanych.
Przy rozpoczynaniu działania makroasemblera, stan odpowiadający tej dyrektywie występuje jako standardowy.

7.3.5.10. .LALL - pełne listowanie rozwinięć makroinstrukcji i powtarzanych bloków

Postać:

.LALL

Dyrektywa ta ustala wyprowadzanie na listing rozwinięć makroinstrukcji i powtarzanych bloków, łącznie z fragmentami warunkowo nieasemblowanymi.

7.3.5.11. .SALL - wstrzymanie listowania rozwinięć

Postać:

.SALL

Dyrektywa wstrzymuje wyprowadzenie na listing rozwinięć makroinstrukcji i powtarzanych bloków.

7.3.5.12. .XCREF i CREF - wyłączenie i włączenie generowania tablicy odwołań

Postać:

.XCREF - wyłączenie generowania tablicy odwołań

.CREF - włączanie generowania tablicy odwołań

Dyrektywy służą do włączania i wyłączania generowania tablicy odwołań. Tablica odwołań przygotowywana jest tylko wówczas, gdy przy wywoływaniu makroasemblera użyto parametru /C/ ptk. 6.1/.
Jeżeli parametru tego nie użyto, to dyrektywy .XCREF i .CREF nie wpływają na pracę makroasemblera.

Dyrektywy .XCREF i .CREF służą do selektywnego wybrania tych fragmentów programu źródłowego, dla których ma być utworzona tablica odwołań /pkt. 6.3/.

7.3.6. Dyrektywy dla makroinstrukcji i powtórzeń bloku

Makroinstrukcje pozwalają definiować bloki złożone z instrukcji makroasemblera, które mogą być replikowane bez potrzeby ponownego ich wypisywania.
Podobnie można stosować wielokrotne powtarzanie bloków przy użyciu specjalnych dyrektyw powtórzeń.

W makroinstrukcjach są stosowane specjalne, niżej podane oznaczenia:

8 - wykorzystywany w rozwinięciu makroinstrukcji do realizacji konkatenacji tekstów lub symboli. Jeśli ma nastąpić podstawienie dla parametru formalnego w postaci łańcucha znaków w cudzysłowiu, wówczas łańcuch ten

powinien mieć na pierwszej pozycji umieszczony znak 8.

;; - jeżeli w bloku komentarz poprzedzony jest dwoma znakami średników /a nie tylko jednym/, to nie jest on uwzględniany w czasie asemblacji przy rozwijaniu tego bloku, tj. nie pojawia się w części listingu przedstawiającej rozwinięcia.

! - wykrzyknik jest stosowany dla pokazania, że następny znak należy traktować dosłownie, tj. jako określona reprezentacja kodu.

% - znak procentu jest stosowany przy wywoływaniu makroinstrukcji, do napisania przed wybranym parametrem aktualnym /wyrażeniem/.
Zapis taki powoduje zamianę tego parametru na liczbę przedstawiającą jego wartość, według bieżącego sposobu przedstawienia liczb. Liczba otrzymana jest podstawiona w miejscu parametru formalnego.

7.3.6.1. MACRO i ENDM - definiowanie makroinstrukcji

Postać:

```
nazwa MACRO parametr formalny  parametr formalny ...
          .
          .
          .
          ENDM
```

Dyrektywa MACRO służy do zdefiniowania ciągu instrukcji zakończonych dyrektywą ENDM.
Ciąg tych instrukcji stanowi ciało makroinstrukcji.
Nazwa poprzedzająca dyrektywę MACRO stanowi nazwę makroinstrukcji.
Nazwa ta może być dowolnej długości lecz tylko 16 pierwszych znaków jest uwzględnianych.
Nazwa ta występuje podobnie do etykiety z tym, że nie występuje po niej znak dwukropka.
Po zdefiniowaniu makroinstrukcji nazwa tej makroinstrukcji służy do jej wywoływania.
Parametr formalny makroinstrukcji może zawierać do 32 znaków i przy wywoływaniu makroinstrukcji jest zamieniany na odpowiedni parametr aktualny, według kolejności występowania na liście parametrów.
Liczba parametrów formalnych jest ograniczona tylko przez długość wiersza /programu źródłowego/.

Postać wywołania makroinstrukcji jest następująca:

nazwa parametr aktualny parametr aktualny ...

Liczba parametrów aktualnych w wywołaniu makroinstrukcji nie musi być równa liczbie parametrów formalnych.
Jeżeli parametrów aktualnych jest więcje, to nadmiarowe parametry są ignorowane.
Jeśli parametrów aktualnych jest mniej, to pozostałe parametry na liście są traktowane jako nie występujące.

Przy ujęciu w nawiasy ostrokątne kilku parametrów z listy parametrów w czasie wywoływania makroinstrukcji, cała zawartość tego nawiasu jest przekazywana jako jeden parametr.

Makrodefinicja przy asemblacji nie jest translowana. Przy wywoływaniach makroinstrukcji, makroasembler zastępuje to wywołanie ciałem makroinstrukcji i asembluje otrzymane zdania.

7.3.6.2. LOCAL - deklaracja symboli wewnętrznych makroinstrukcji

Postać:

LOCAL parametr formalny parametr formalny ...

Dyrektywa ta może występować tylko wewnątrz definicji makroinstrukcji i jako pierwsze instrukcje ciała makroinstrukcji.
W czasie asemblacji makroasembler wytwarza określony, niepowtarzalny symbol dla każdego parametru formalnego w postaci od ..0001 do ..FFFF.
Powyższe symbole parametrów formalnych są zwykle wykorzystywane jako etykiety wewnątrz makroinstrukcji.
W ten sposób unika się wielokrotnego definiowania etykiet, które może powstać przy kolejnych wywołaniach tej samej makroinstrukcji.
Użytkownik powinien unikać stosowania własnych etykiet o postaci ...nnnn, gdzie n jest dowolną cyfrą heksadecymalną, celem wyeliminowania niejednoznaczności.

7.3.6.3. REPT i ENDM - powtórzenie bloku

Postać:

```
REPT  wyrażenie
  .
  .
  .
ENDM
```

Dyrektywa ta powoduje powtórzenie bloku instrukcji zawartych między REPT a ENDM, tyle razy, ile wynosi wartość wyrażenia.

Wyrażenie to jest określone jako 16-to bitowa liczba bez znaku.

Jeżeli wyrażenie zawiera symbol zewnętrzny lub niezdefiniowany operand, to sygnalizowany jest błąd.

Należy zwrócić uwagę, że dyrektywa ENDM, oznaczająca koniec bloku zdań podlegającego powtórzeniu, jest taka sama, jak dyrektywa oznaczająca koniec ciągu zdań w definicji makroinstrukcji.

7.3.6.4. IPR i ENDM - powtórzenie bloku według listy parametrów

Postać:

```
IRP   parametr formalny ,  arglist
 .
 .
 .
ENDM
```

Arglist stanowi listę parametrów /symboli łańcuchów znaków, stałych liczbowych lub znakowych/ oddzielonych przecinkami i ujętych łącznie w nawiasy ostrokątne.

Dyrektywa IPR powoduje powtórzenie bloku instrukcji zawartych między IPR i ENDM, tyle razy, ile parametrów występuje w nawiasie ostrokątnym.

Przy każdym powtórzeniu w miejscu parametru formalnego jest podstawiany kolejny parametr z listy parametrów.

7.3.6.5. IRPC i ENDM - powtórzenie bloku według łańcucha znaków

Postać:

```
IRPC   parametr formalny,  łańcuch znaków
  .
  .
ENDM
```

Dyrektywa ta powoduje powtórzenie bloku zdań zawartych między IRPC i ENDM, tyle razy, ile znaków występuje w łańcuchu znaków.
Przy każdym powtórzeniu w miejsce parametru formalnego jest podstawiany kolejny znak brany z łańcucha znaków.

7.3.6.6. EXITM - wyjście z makroinstrukcji

Postać:

```
EXITM
```

Dyrektywa ta stosowana jest wewnątrz makroinstrukcji lub powtarzanego bloku /dyrektywy REPT, IRP, IRPC/ do przerwania rozwijania, gdy wystąpią warunki określające rozwijanie dolne jako niepotrzebne lub niepożądane.
Zwykle EXITM jest używana łącznie z dyrektywą warunkowej asemblacji.
Jeżeli następuje asemblacja EXITM, wówczas przerwane jest dalsze rozwijanie. Jeżeli blok zawierający EXITM jest zagnieżdżony wewnątrz innego bloku, wówczas rozwijanie bloku wewnętrznego po przerwaniu

rozwijania bloku wewnętrznego jest kontynuowane.

7.3.7. Dyrektywy asemblacji warunkowej

Dyrektywy asemblacji warunkowej pozwalają wyznaczyć wybrane bloki instrukcji programu źródłowego, które są albo asemblowane, albo pomijane przy asemblacji, w zależności od spełnienia określonych warunków, sprawdzanych przez te dyrektywy.

Warunkowa asemblacja może być zagnieżdżona do 255 poziomów.
Argument określający warunek asemblacji powinien być zdefiniowany już w pierwszym przebiegu asemblacji gdyż w przeciwnym przypadku występuje błędne działanie i sygnalizowany jest błąd V.

Dla dyrektyw IF/IFT/COND i IFF/IFE wyrażenia powinny być absolutne.

Każdy blok asemblacji warunkowej może zawierać opcjonalną dyrektywę ELSE, wyznaczającą zmienną część bloku podlegającą asemblacji, gdy nie jest spełniony warunek asemblacji.

Tylko jedna dyrektywa ELSE może wystąpić dla danego bloku asemblacji warunkowej. ELSE jest powiązane z najbliżej położoną dyrektywą badającą warunek asemblacji.
Wystąpienie większej niż dopuszczalna ilość ELSE powoduje sygnalizację błędu C.

7.3.7.1. Dyrektywy badające warunek asemblacji

a/ Postać:

```
IF      wyrażenie
IFT     wyrażenie
COND    wyrażenie    /przy trybie pracy .Z80/
```

Jeśli wartość wyrażenia jest różna od zera, to instrukcje wewnątrz bloku asemblacji warunkowej są asemblowane.

b/ Postać:

```
IFE     wyrażenie
IFF     wyrażenie
```

Jeśli wartość wyrażenia jest równa zero, to instrukcje wewnątrz bloku asemblacji warunkowej są asemblowane.

c/ Postać:

```
IF1
```

Jeśli występuje pierwszy przebieg asemblacji, to instrukcje wewnątrz bloku asemblacji warunkowej są asemblowane.

d/ Postać:

```
IF2
```

Jeśli występuje drugi przebieg asemblacji, to instrukcje wewnątrz bloku asemblacji warunkowej są asemblowane.

e/ Postać:

IFDEF symbol

Jeśli symbol jest zdefiniowany lub zadeklarowany jako zewnętrzny, to instrukcje wewnątrz bloku asemblacji warunkowej są asemblowane.

f/ Postać:

IFNDEF symbol

Jeśli symbol nie jest zdefiniowany lub nie jest zadeklarowany jako zewnętrzny, to instrukcje wewnątrz bloku asemblacji warunkowej są asemblowane.

g/ Postać:

IFB argument

Wymagany jest zapis nawiasów ostrokątnych razem z argumentem.
Jeśli nie występuje zapis argumentu /tj. brak jest zawartości między ostrokątnymi nawiasami lub jakiegokolwiek zapisu poza nazwą dyrektywy/, to zdania wewnątrz bloku asemblacji warunkowej są asemblowane.

h/ Postać:

IFNB argument

Wymagane jest zapis nawiasów ostrokątnych razem z argumentem.

Jeśli występuje zapis argumentu, to zdania wewnątrz bloku asemblacji warunkowej są asemblowane.

Dyrektywa stosowana jest dla testowania parametrów formalnych.

i/ Postać:

IFIDN argument 1 , argument 2

Wymagany jest zapis nawiasów ostrokątnych razem z argumentami.
Jeśli łańcuch argument 1 jest identyczny z łańcuchem argument 2, to zdania wewnątrz bloku asemblacji warunkowej są asemblowane.

j/ Postać:

IFDIF argument 1 , argument 2

Wymagany jest zapis nawiasów ostrokątnych razem z argumentami.
Jeśli łańcuch argument 1 różni się od łańcucha argument 2, to zdania wewnątrz bloku asemblacji warunkowej są asemblowane.

7.3.7.2. ELSE - określenie zamiennego kodu przy asemblacji

Postać:

ELSE

Pomocnicza dyrektywa stosowana opcjonalnie, wydzielająca w bloku asemblacji warunkowej tę część bloku, która jest asemblowana, jeśli początkowa część całego bloku nie jest asemblowana.
Może być stosowana w połączeniu z dowolną dyrektywą badającą warunek asemblacji.

7.3.7.3. ENDIF / ENDC - koniec bloku warunkowej asemblacji

Postać:

ENDIF

ENDC /przy trybie pracy .Z80/

Powyższe dyrektywy wyznaczają koniec bloku warunkowej asemblacji.
Powinny one wystąpić dla każdej dyrektywy badającej warunek asemblacji.
Dla dyrektywy badającej warunek typu IFxxx, należy stosować dyrektywę kończącą ENDIF.

Dla dyrektywy badającej warunek COND, należy stosowac dyrektywę kończącą ENDC.

8. MAKROASEMBLER - MAC

Makroasembler - MAC służy do translacji programów napisanych w języku asemblera INTEL 8080 lub Z80 na przesuwalny kod wynikowy.

Wywołanie programu jest następujące:

MAC nazwa programu

gdzie "nazwa programu" określa nazwę pliku kodów **asemblera** o typie /rozszerzeniu/ "ASM".

Na zakończenie procesu translacji MAC tworzy plik "nazwa programu". HEX zawierający kody wynikowe w postaci heksadecymalnej, który można załadować i testować programami SID i ZSID /patrz dodatek D/ lub ładować i uruchamiać zleceniem LOAD /patrz "CP/J. System operacyjny CP/J - dla mikrokomputera ELWRO 800-JUNIOR"/.

Dodatkowo zostaje utworzony plik o nazwie "nazwa programu .PRN" zawierający listing programu źródłowego wraz z komentarzami i wynikiem translacji, oraz plik "nazwa programu .SYM" zawierający listę symboli zdefiniowanych w programie.
Postać programu źródłowego, instrukcje języka oraz następujące dyrektywy makroasemblera: ORG, END, EQU, SET, IF, ELSE, ENDIF, DB, DW, DS, PAGE i TITLE, które mogą być stosowane w programach translowanych przez makroasembler MAC opisano w rozdziale 7 tego podręcznika.

Makroasembler MAC może być wywoływany w sposób opisany wyżej bez parametrów lub z parametrami.

Wywołanie programu z parametrami jest następujące:

MAC nazwa programu parametry

Lista parametrów:

A - określenie dysku źródłowego dla pliku.ASM

H - określenie dysku dla pliku .HEX

L - określenie dysku dla pliku .LIB

M - kontrola listingu MACRO w pliku .PRN

P - określenie dysku dla pliku .PRN zawierającego listing

Q - kontrola listingu symboli LOCAL

S - określenie generacji i dysku dla pliku .SYM

1 - kontrola listingu i przebiegu.

W przypadku parametrów A, H, L i S następuje po nich nazwa dysku, z którego dane są czytane lub, na który mają być wyprowadzone przy czym jednostki dyskowe mogą mieć nazwy A, ..., Z.

Przyjęto konwencję, że urządzenie X odpowiada konsoli użytkownika, P jest systemową drukarką /logicznie urządzenie listujące LST:/ a Z plik zerowy, który nie jest tworzony na dysku.

Po wywołaniu MAC i nazwie pliku źródłowego następuje

[illegible] X PB A HB SX

co powoduje zapisanie pliku PRN na dysk B i wczytanie pliku ASM z dysku A po czym kieruje plik HEX na dysk B i wysyła plik SYM na konsolę użytkownika.

Jeśli parametry L, S, M, Q, 1 będą poprzedzone znakiem "+" lub "-" otrzymuje się, wówczas następujące modyfikacje funkcji:

+ L	listowanie informacji czytanej z makro biblioteki
- L	opuszczenie listingu makro biblioteki
+ S	dodanie pliku .SYM na końcu pliku .PRN
- S	pominięcie generacji tabeli symboli
+ M	listing wszystkich linii makro przetworzonych w czasie translacji
- M	pominięcie wszystkich linii makro przetworzonych w czasie translacji
M	listing tylko "hex" generowany przez rozszerzenie makro
Q	listing wszystkich symboli LOCAL z listy symboli
- Q	pominięcie symboli LOCAL z listy symboli
+ 1	utworzenie pliku listingu dla pierwszego przebiegu

- 1 pominięcie listingu przy pierwszym przebiegu

Przykładowe poniższe parametry translacji:

¤PB + S - M HB

powodują zapisanie pliku PRN na dysku B razem z listą symboli.
Wszystkie generacje makro zostaną pominięte, a plik HEX zostaje zapisany wraz z plikiem PRN na dysku B.

Zakładając, że dysk b jest dyskiem bieżącym makroasembler MAC zakłada następujące parametry: pliki ASM i LIB zostają przydzielone do dysku b, a pliki HEX, PRN i SYM zostają również wysłane na dysk b. Tablica symboli generowana jest z pominięciem symboli LOCAL i listowane są linie makro generujące kod wynikowy.

Przykłady:

Plik przeznaczony do translacji nazywa się "ZADANIE", przykładowe wywołanie MAC może mieć postać:

MAC ZADANIE ¤PX + S - M

które transluje plik ZADANIE .ASM z listowaniem na konsoli, ale bez listingu generowanych makroinstrukcji.

MAC A : ZADANIE ¤+S - m + q

transluje plik ZADANIE .ASM z dysku A, tworzy ZADANIE .PRN na dysku bieżącym, pomijając generowanie makro oraz oprócz normalnie listowanych symboli listuje symbole rozpoczynające się od znaku "??" /etykiety typu LOCAL/.

MAC ZADANIE

transluje plik ZADANIE .ASM z dysku bieżącego, tworzy plik ZADANIE .PRN, ZADANIE .SYM i plik ZADANIE .HEX na tym samym dysku.
Można dodatkowo stosować poniższe funkcje kontrolne umieszczone w tekście programu i umieszczone na początku dowolnej linii tekstu programu:

¤ - PRINT	zatrzymanie generowania listingu
¤ - PRINT	umożliwienie poprzednio wstrzymanego drukowania listingu
¤ - MACRO	pominięcie generowania makro linii w listingu
¤ - MACRO	wprowadzenie wszystkich form makro
¤ * MACRO	wprowadzenie części form makro /jak w "*M"/.

Ponieważ makroasembler MAC umożliwia wprowadzenie numeru każdej linii, znak kontrolny " ¤ " powinien być umieszczony zaraz po numerze linii.

ZESTAWIENIE DYREKTYW MAKROASEMBLERA M80

W zestawieniu dyrektywy, odpowiedające dotychczas używanym dla mikroprocesora Z80, oznaczono gwiazdką /*/, a stosowane dla mikroprocesora 8080 pozostawiono bez gwiazdki. Makroasembler MAKREL akceptuje /jeśli nie podano inaczej/ wszystkie dyrektywy dla obu typów mikroprocesora.

Dyrektywy wyboru zestawu instrukcji

.Z80

.8080

Dyrektywy definiowania danych i symboli

nazwa ASET wyrażenie

BYTE EXT symbol

BYTE EXTRN symbol

BYTE EXTERNAL symbol

DB wyrażenie wyrażenie ...

DB łańcuch łańcuch ...

DC łańcuch

DDB wyrażenie wyrażenie ...

DEFB wyrażenie wyrażenie ...

nazwa DEFL wyrażenie

DEFM łańcuch łańcuch ...

DEFS wyrażenie wartość ...

DEFW wyrażenie wyrażenie ...

DS wyrażenie wartość

DW wyrażenie wyrażenie ...

ENTRY nazwa nazwa ...

nazwa EQU wyrażenie

EXT nazwa nazwa ...

EXTRN nazwa nazwa ...

EXTERNAL nazwa nazwa ...

GLOBAL nazwa nazwa ...

PUBLIC nazwa nazwa ...

nazwa SET wyrażenie /nie używana w trybie pracy .Z80/

Dyrektywy trybu asemblacji

ASEG

CSEG

DSEG

COMMON nazwa bloku

ORG wyrażenie

.PHASE wyrażenie

.DEPHASE

Dyrektywy dotyczące pliku z programem źródłowym

.COMMENT ogranicznik tekst ogranicznik

END wyrażenie

INCLUDE nazwa pliku

$INCLUDE nazwa pliku

MACLIB nazwa pliku

NAME / nazwa modułu

.RADIX wyrażenie

.REQUEST nazwa pliku, nazwa pliku ...

Dyrektywy sterujące dla listingu

sterowanie formatem

EJECT wyrażenie /gwiazdka w nazwie dyrektywy/

PAGE wyrażenie

SUBTTL tekst

TITLE tekst

$TITLE tekst

dyrektywy ogólnego przeznaczenia

.LIST

.XLIST

.PRINTX ogranicznik tekst ogranicznik

sterowanie wydrukiem warunkowej asemblacji

.SFCOND
.LFCOND
.TFCOND

sterowanie wydrukiem dla rozwinięć

.LALL

.SALL

.XALL

sterowanie wydrukiem odwołań

.XCREF

.CREF

Dyrektywy dla makroinstrukcji i powtórzeń bloku

dla makroinstrukcji

nazwa MACRO parametr parametr ...

ENDM

EXITM

LOCAL parametr parametr ...

dla powtórzeń

REPT wyrażenie

IRP parametr formalny parametr w nawiasach ostrokąt.

IRPC parametr formalny łańcuch

Dyrektywy asemblacji warunkowej

COND wyrażenie
ELSE
ENDC
ENDIF
IF wyrażenie

```
IFB   argument

IFDEF   symbol

IFDIF   agrument 1   argument 2

IFE   wyrażenie

IFF   wyrażenie

IFIDN   argument 1   argument 2

IFNB   argument

IFNDEF   symbol

IFT   wyrażenie

IF1

IF2
```

ZESTAWY INSTRUKCJI DLA MIKROPROCESORÓW Z80 I 8080

Instrukcje dla Z80

Kod operacji	Funckja
ADC A	Dodanie z przeniesieniem do akumulatora
ADC HL, rp	Dodanie pary rejestrów z przeniesieniem do HL
ADD	Dodanie
AND	Iloczyn logiczny
BIT	Testuj
CALL adres	Wywołanie podprogramu
CALL war, adres	Wywołanie warunkowe
CCF	Zmiana wskaźnika przeniesienia
CP	Porównanie
CPD	Porównanie i zmniejszenie
CPDR	Porównanie, zmniejszenie i powtórzenie
CPI	Porównanie i zwiększenie
CPIR	Porównanie, zwiększenie i powtórzenie
CPL	Dopełnienie akumulatora
DAA	Przystosowanie akumulatora do liczb dziesiętnych
DEC	Zmniejszenie
DI	Zakaz przerwań
DJNZ	Zmniejszenie i skok, jeśli nie zero

EI	Umożliwienie przerwań
EX	Wymiana
EXX	Wymiana par rejestrów i alternatywy
HALT	Stop
IM x	Ustalenie trybu przerwań
IN	Wejście
INC	Zwiększenie
IND	Wejście i zmniejszenie
INDR	Wejście, zmniejszenie i powtórzenie
INI	Wejście i zwiększenie
INIR	Wejście, zwiększenie i powtórzenie
JP adres	Skok
JP war, adres	Skok warunkowy
JR	Skok względny
JR war, adres	Skok względny, warunkowy
LD A, /adres/	Ładowanie akumulatora
LD A, /BC/ lub /DE/	Ładowanie akumulatora pośrednie
LD A, I	Ładowanie akumulatora z rejestru wektora przerwania
LD A, R	Ładowanie akumulatora z rejestru odświeżania
LD HL, /adres/	Ładowanie bezpośrednie HL
LD dane	Ładowanie natychmiastowe
LD xy, /adres/	Ładowanie bezpośrednie rejestru indeksu
LD rejestr /HL/	Ładowanie rejestru
LD rejestr /xy+disp/	Ładowanie rejestru z indeksowaniem
LD rp, /adres/	Ładowanie pary rejestrów
LD SP, HL	Przesłanie HL do wskaźnika stron
LD SP, xy	Przesłanie rejestru indeksu do wskaźnika stron
LD przeznaczenie, źródło	Przesłanie rejestr - rejestr
LD /adres/, A	Zapamiętanie akumulatora

LD /BC/ lub /DE/, A	Zapamiętanie akumulatora pośrednie
LD I, A	Zapamiętanie akumulatora w rejestrze wektora przerwania
LD R, A	Zapamiętanie akumulatora w rejestrze odświeżania
LD /HL/, dane	Zapamiętanie natychmiastowe w pamięci
LD /adres/, HL	Zapamiętanie bezpośrednie HL
LD /xy+disp/, dane	Zapamiętanie natychmiastowe w pamięci z indeksowaniem
LD /adres/, xy	Zapamiętanie bezpośrednie rejestru indeksu
LD /HL/, rejestr	Zapamiętanie rejestru
LD /xy+disp/, rejestr	Zapamiętanie rejestru z indeksowaniem
LD /adres/, rp	Zapamiętanie pary rejestrów
LDD	Załadowanie i zmniejszenie
LDDR	Załadowanie, zmniejszenie i powtórzenie
LDI	Załadowanie i zwiększenie
LDIR	Załadowanie, zwiększenie i powtórzenie
NEG	Zaszeregowanie akumulatora
NOP	Nic nie rób
OR	Suma logiczna
OUT	Wyjście
OUTD	Wyjście i zmniejszenie
OTDR	Wyjście, zmniejszenie i powtórzenie
OUTI	Wyjście i zwiększenie
OTIR	Wyjście, zwiększenie i powtórzenie

7110.5022–01 33–0

POP	Pobranie ze stosu
PUSH	Umieszczenie na stosie
RES	Zerowanie bitu
RET	Powrót z podprogramu
RET war	Powrót warunkowy
RETI	Powrót z przerwania
RETN	Powrót z niemaskowanego przerwania
RL	Przesunięcie z przeniesieniem w lewo
RLA	Przesunięcie akumulatora z przeniesieniem w lewo
RIC	Przesunięcie cykliczne e lewo
RICA	Przesunięcie akumulatora cyklicznie w lewo
RLD	Przesunięcie akumulatora i pamięci dziesiętnie w lewo
RR	Przesunięcie z przeniesieniem w prawo
RRA	Przesunięcie akumulatora z przeniesieniem w prawo
RRC	Przesunięcie cykliczne w prawo
RRCA	Przesunięcie akumulatora cyklicznie w prawo
RRD	Przesunięcie akumulatora i pamięci dziesiętnie w prawo
RST	Restart
SET	Ustawienie bitu
SBC	Odejmowanie z pobraniem
SCF	Ustawienie przeniesienia
SLA	Przesunięcie arytmetyczne w lewo
SRA	Przesunięcie arytmetyczne w prawo
SRL	Przesunięcie logiczne w prawo
SUB	Odejmowanie

XOR	Logiczna różnica symetryczna

Instrukcje dla 8080

Kod operacji	Funkcja
ADC, ACI	Dodanie z przeniesieniem
ADD, ADI	Dodanie
ANA, ANI	Iloczyn logiczny
CALL	Wywołanie podprogramu
CC	Wywołanie, jeżeli przeniesienie
CM	Wywołanie, jeżeli minus
CMA	Dopełnienie akumulatora
CMC	Dopełnienie przeniesienia
CMP, CPI	Prównanie
CNC	Wywołanie, jeżeli brak przeniesienia
CNZ	Wywołanie, jeżeli nie zero
CP	Wywołanie, jeżeli plus
CPE	Wywołanie, jeżeli parzystość
CPO	Wywołanie, jeżeli nieparzystość
CZ	Wywołanie, jeżeli zero
DAA	Przystosowanie akumulatora do liczb dziesiętnych
DAD	Dodawanie 16-to bitowe
DCR	Zmniejszenie
DCX	Zmniejszenie dla 16-tu bitów
DI	Zakaz przerwań
EI	Umożliwienie przerwań
HLT	Stop
IN	Wejście
INR	Zwiększenie

INX	Zwiększenie dla 16-tu bitów
JC	Skok, jeżeli przeniesienie
JM	Skok, jeżeli minus
JMP	Skok
JNC	Skok, jeżeli brak przeniesienia
JNZ	Skok, jeżeli nie zero
JP	Skok, jeżeli plus
JPE	Skok, jeżeli parzystość
JPO	Skok, jeżeli nieparzystość
JZ	Skok, jeżeli zero
LDA	Ładowanie akumulatora
LDAX	Ładowanie akumulatora pośrednie
LHLD	Ładowanie bezpośrednie HL
LXI	Ładowanie 16-to bitowe
MOV	Przesłanie
MVI	Przesłanie natychmiastowe
NOP	Nic nie rób
ORA, ORI	Suma logiczna
OUT	Wyjście
PCHL	HL do licznika rozkazów
POP	Pobranie ze stosu
PUSH	Umieszczenie na stosie
RAL	Przesunięcie z przeniesieniem w lewo
RAR	Przesunięcie z przeniesieniem w prawo
RC	Powrót, jeżeli przeniesienie
RET	Powtór z podprogramu
RLC	Przesunięcie w lewo
RM	Powrót, jeżeli minus
RNC	Powrót, jeżeli brak przeniesienia
RNZ	Powrót, jeżeli nie zero
RP	Powrót, jeżeli plus
RPE	Powrót, jeżeli parzystość

RPO	Powrót, jeżeli nieparzystość
RRC	Przesunięcie w prawo
RST	Restart
RZ	Powrót, jeżeli zero
SBB, SBI	Odejmowanie z pobraniem
SHLD	Zapamiętanie bezpośrednio HL
SPHL	HL do wskaźnika stosu
STA	Zapamiętanie akumulatora
STAX	Zapamiętanie akumulatora pośrednio
STC	Ustawienie przeniesienia
SUB, SUI	Odjęcie
XCHG	Zamiana DE i HL
XRA, XRI	Logiczna różnica symetryczna
XTHL	Zamiana szczytu stanu z HL

PROGRAM ŁĄCZĄCY - L80

Załącznik C zawiera opis i sposób uruchomienia programu L80. Program przeznaczony jest dla użytkowników i programistów pracujących na mikrokomputerze ELWRO 800-JUNIOR wyposażonym w system operacyjny CP/J.

Służy do ładowania i łączenia programów typu REL w plik zawierający program wynikowy.

1. Uruchomienie programu L80

Do uruchomienia programu L80 służy komenda:

L80

po której zostaje wyprowadzony znak " * " wskazujący gotowość do przyjmowania komend.

2. Komendy programu L80

Komendy programu L80 mogą być w postaci:

- nazw plików oddzielonych przecinkami
- dyrektyw

Komendy można wprowadzać pojedyńczo / po znaku " * "/ lub wszystkie w jednej linii /łącznie z L80/.

Przykład 1

```
L80              (CR)
 *nazwa pliku, nazwa pliku   (CR)
 */dyrektywa
```

Przykład 2

L80 nazwa pliku, nazwa pliku /dyrektywa ... (CR)

Wywoływanie pojedyńczych zleceń jest wolniejsze w działaniu, ale pozwala użytkownikowi śledzić kolejność wykonywanych czynności.

2.1. Nazwa pliku jako zlecenie jest określeniem pliku zgodnym ze standardem stosowanym w systamie operacyjnym CP/J.

2.2. Dyrektywy

W łańcuchu komend programu L80 można podać szereg dyrektyw służących do określenia działań wpływających na ładowanie lub łączenie programu.
Każda dyrektywa musi być poprzedzona znakiem "/".
W linii zlecenia można zapisać dowolną ilość dyrektyw.
Umieszczenie dyrektywy na początku w łańcuchu zlecenia odnosi się do programów już załadowanych do pamięci.
Umieszczenie dyrektywy bezpośrednio za nazwą pliku powoduje działanie dyrektywy tylko dla tego pliku.
Dyrektywy umieszczone na końcu łańcucha komend wpływają na cały proces ładowania i łączenia plików /programów/.

2.2.1. Opis dyrektyw

/N

Dyrektywa powoduje zapisanie połączonego programu do pliku wynikowego.
Dyrektywa ta musi występować bezpośrednio po nazwie pliku do którego będzie zapisany program wynikowy.
Jeżeli nie poda się typu pliku wynikowego to domyślnym typem jest COM.

Należy zawsze podać przynajmniej jeden plik wynikowy.

/E

Dyrektywa ta powoduje załadowanie do pamięci programu, którego nazwa została określona w linii zlecenia, wykonanie łączenia i powrót do systemu operacyjnego CP/J.
Po zakończeniu działania L80 wyprowadza informację o zajętości pamięci /patrz dyrektywa /G /.

Przykład:

L80 ZB1, ZB1.COM /N/E

gdzie:

ZB1 - nazwa pliku wejściowego z rozszerzeniem REL

ZB1.COM - nazwa i rozszerzenie pliku wynikowego

L80 ZB1.COM/N, ZB1, ZB2, ZB3/E

ZB1.COM - nazwa pliku wynikowego
ZB1, ZB2, ZB3 - nazwy plików wejściowych /z rozszerzeniem REL/

Aby uruchomić zachowany na dysku program wynikowy, należy podać, na poziomie zleceń nierezydentnych systemu nazwę programu.

/G

Dyrektywa ta powoduje załadowanie do pamięci programu określonego w linii zlecenia , wykonanie łączenia i uruchomienie utworzonego programu wynikowego.
Po zakończeniu wykonania programu następuje powrót do systemu operacyjnego CP/J.

Przykład:

L80 JX, JX/N/G

Po połączeniu pliku JX.REL i utworzeniu pliku JX.COM nastąpi uruchomienie i wykonanie programu JX.COM oraz powrót do systemu operacyjnego CP/J.

Przed uruchomieniem programu użytkowego , L80 wyprowadza informację określającą adres początkowy programu, adres ostatniego bajtu programu /adres końcowy/, ilość stron /256-bajtowych/ zajętych przez program od adresu 100H do adresu końca programu.

Przykład:

/4010 41F1 65/

gdzie:

4010 - adres początkowy programu

41F1 - adres końcowy programu

65 - ilość stron

UWAGI:

Dyrektywa /N nie działa, jeżeli nie wystąpi po niej dyrektywa /E lub /G. Należy ją stosować zawsze gdy musi być zachowany program wynikowy. Użycie dyrektyw /N/E lub /N/G jest wymagane w większości operacji realizowanych przez program L80. Dyrektywy /N/E powodują łączenie plików /programów/ w jeden plik /program/ wynikowy, uruchomienie utworzonego programu wynikowego i powrót do systemu operacyjnego CP/J. Jeżeli używa się dyrektywy /G program zostaje automatycznie uruchomiony bez względu na rozszerzenie nazwy programu wynikowego.

/P: adres

Dyrektywa służy do ustawienia adresu początkowego programu i danych.
Wartość adresu musi być wyrażona w aktualnym systemie liczenia, /domyślnie przyjmowany jest system szesnastkowy/.
Jeżeli nie poda się parametru adres , to przyjmowany jest adres 103H.
Dyrektywa /P pozwala umieścić segmenty programu pod różne adresy. Należy ją umieścić przed nazwą pliku, który ma być ładowany począwszy od określonego adresu np.:

L80 /P:103,IX,IX/N/E

W czasie jednego przebiegu łączenia można użyć kilku dyrektyw /P do umieszczenia różnych segmentów pod różnymi adresami. Należy jednak zwrócić uwagę, aby obszary segmentów nie nakładały się, obszary programu nie mogą przeplatać się z obszarami danych i COMMON.
W segmencie z programem mogą być przerwy między modułami, ale segment ten nie może zawierać danych i odwrotnie.

Program L80 nie wypełnia przerw spacjami.

/D: <adres>

Dyrektywa działa tak samo jak dyrektywa /P /dla obszarów danych i COMMON/.
Użycie dyrektyw /P i /D powoduje, że obszary danych i COMMON będą załadowane pod adres określony w /D, a początek obszaru programu zostanie określony przez adres podany w /P.

/R

Dyrektywa ta powoduje powrót programu L80 do poziomu zleceń operatora.
Po napotkaniu tej dyrektywy program L80 przerywa obsługę ostatniej linii zlecenia, ignoruje pliki, wyprowadza na monitor ekranowy znak " " i oczekuje na wprowadzenie nowego zlecenia.

/U

Dyrektywa powoduje wylistowanie wszystkich niezdefiniowanych symboli globalnych.
Działa tylko wtedy, gdy w linii zlecenia nie występują dyrektywy /G i /E.
Listowanie symboli można przerwać używając CTRL-S oraz wznowić przez CTRL-Q.

/M

Dyrektywa powoduje wyprowadzenie na monitor zdefiniowanych oraz niezdefiniowanych symboli globalnych.

/Q

Dyrektywa zmienia system liczenia na ósemkowy.

/H

Dyrektywa zmienia system liczenia na szesnastkowy.

/X

Dyrektywa ta powoduje utworzenie pliku wynikowego w szesnastkowym kodzie ASCII.
Dyrektywa musi być połączona z dyrektywą /N i /E.
Plik zachowany przy pomocy dyrektyw /X, /N i /E otrzymuje rozszerzenie HEX.
Dyrektywa /X stosowana jest głównie w celu przygotowania programów do zapisu w pamięci ROM.

/Y

Dyrektywa powoduje utworzenie dodatkowo pliku wynikowego w specjalnym formacie typu SYM.
Plik SYM zawiera nazwy i adresy wszystkich symboli globalnych. Pozwala to na użycie programów SID i ZSID.

Komunikaty o błędach

Błędy powstałe podczas uruchomienia programu L80 sygnalizowane są przez odpowiednie komunikaty a nie kod błędu.

Większość komunikatów zaczyna się symbolem "?" lub "%".

? Brak adresu startu - została wydana dyrektywa /G lecz nie został załadowany żaden program główny

? Błąd odczytu - ostatni plik opisany jako wyjściowy jest źle sformatowany lub brak odpowiedniej ilości nazw plików

? Brak pamięci - brak dostatecznej ilości pamięci do załadowania programu

? Błędna komenda	-	została wydana nieprawidłowa komenda programu ELINK
? nazwa pliku brak	-	plik o nazwie nazwa pliku nie został znaleziony na dysku
? Etykiety startu - nieznane - brak def.	-	po podaniu dyrektywy /E lub /G niezdefiniowany by- symbol adresu startowego
Drugi COMMON większy xxxxxx	-	brak miejsca do alokacji segmentu wspólnego COMMON tzn. że segment wspólny o nazwie xxxxxx w pierwszym z modułów nie był największym blokiem. Należy zmienić kolejność ładowania modułów lub zmienić definicję bloku COMMON tak, aby wszystkie bloki miały tę samą wielkość
Wielokrotnie zdefiniowana zmienna globalna yyyyyy	-	jeden symbol globalny o nazwie yyyyyy posiada więcej niż jedną definicję
Nałożenie obszaru danych	,	start = xxxx
	,	Public = nazwa symbolu xxxx
	,	External = nazwa symbolu xxxx
lub		
Nałożenie obszaru kodu	,	start = xxxx
	,	Public = nazwa symbolu xxxx
	,	External = nazwa symbolu xxxx
	-	adresy ustalone przez dyrektywy /P lub /D są zbyt blisko siebie albo zachodzą na siebie. W wyniku tego dane albo program mogą nałożyć się na siebie /poprzednio załadowane moduły mogą się zniszczyć/.

Ingerencja obszaru danych

lub

Ingerencja obszaru kodu - przecinanie się obszarów danych i programu

Lokacja powyżej dostępnej pamięci - zmienić T/N ?

? źródło poniżej , zreorganizować pamięć /Y lub N/? - komunikat pojawia się podczas wykonywania dyrektywy /E lub /G, kiedy adres ładowania był poniżej lub powyżej obszaru pamięci dostępnego dla programu L80 .
Odpowiedź na komunikat ze znakiem Y powoduje automatyczne przemieszczenie segmentów, odpowiedź N powoduje zakończenie działania programu L80 .

? Błąd zapisu pliku COM - błąd na dysku podczas wykonywania operacji zapisu pliku wynikowego

lub

? Błąd zapisu pliku

Brak pliku - brak podanego zbioru wejściowego na dysku

Błąd odczytu - błąd czytania zbioru wejściowego

Dysketka pełna

Start programu

Wolny obszar

? Etykiety startu

DEBUGGER'Y SYMBOLICZNE SID i ZSID

Dodatek ten zawiera opis debugger'ów symbolicznych SID i ZSID. Debugger SID stosowany jest do testowania i uruchamiania programów napisanych w j.asemblera procesora INTEL, ZSID w j. asemblera procesora Z80.

Ponieważ opisy powyższych debugger'ów są takie same, w dodatku opisano debugger używając symbolu "SID", który oznacza zarówno symbol "SID" i "ZSID".

1. Sposoby uruchomienia programu

SID - załadowanie SID bez ładowania programu

SID X,Y - załadowanie SID i programu X, Y

SID X,Y U.V - załadowanie SID, programu X, Y i tabeli symboli U.V

SID U.V - załadowanie SID i tabeli symboli

Program SID po załadowaniu zgłasza się znakiem # .

2. Dyrektywy

2.1. Znaki specjalne

CTRL - C - Powrót do systemu operacyjnego CP/J

CTRL - E - Koniec linii

CTRL - P - Włączenie wyłączenie drukarki

CTRL - R	- przepisanie bieżącej linii
CTRL - S	- zatrzymanie/wznowienie wydruku na konsoli
CTRL - U	- likwidacja linii
CTRL - H	- likwidacja znaku i cofanie kursora
BS	- likwidacja znaku

W każdej chwili wydruk na konsolę/Np. DYREKTYWA D/ może być przerwany przez wciśnięcie dowolnego klawisza.

2.2. Wyrażenia symboliczne

Wszystkie parametry dyrektywy mogą być w postaci wyrażenia.

Wyrażeniem jest dowolny ciąg elementów połączony operatorami dodawania + lub odjemowania - .

Elementami są:

- liczby heksadecymalne

  1 lub więcej cyfr heksadecymalnych - rozpatrywane są tylko 4 ostatnie

  np.
  1
  38001 - rozpatrywane 8001

- liczba dziesiętna

  jeden lub więcej cyfr dziesiętnych rozpoczynających się znakiem .

  np.
  #1
  #256

- znaki ASCII

  1 lub więcej znaków umieszczonych między apostrofami /jeśli znakiem jest apostrof to należy go napisać jako dwa apostrofy/.

  Rozpatrywane są kody dwóch ostatnich znaków.

  np.

'A'	=	0041
'AB'	=	4142
''''	=	0060

- symbole

  Formaty:

.S	-	bezpośrednia wartość symbolu S
ϱ S	-	zawartość dwóch sąsiednich bajtów /wartość 16-bitowa/ wskazywanych przez symbol S.
=S	-	zawartość bajtu wskazywanego przez S.

  np.

  Wartość symbolu GAMMA = 100

  Zawartość komórek

  100 = 02
  101 = 3E

  Wartość elementów wyrażenia

  .GAMMA = 0100
  ϱ GAMMA = 3E02
  =GAMMA = 0002

2.3. Opis dyrektyw

W programach SID i ZSID mogą być używane następujące dyrektywy:

A	- Asemblacja
C	- Wywołanie podprogramu
D	- Wyprowadzenie zawartości pamięci w formacie HEX i ASCII
F	- Wypełnienie pola pamięci stałą wartością
G	- Uruchomienie testowanego programu
H	- Obliczanie wartości heksadecymalnej
I	- Wprowadzenie wiersza
L	- Listowanie instrukcji w postaci mnemonicznej
M	- Przesunięcie bloku pamięci
P	- Ustawienie stałej pułapki
R	- Wczytanie programu lub tablicy symboli
S	- Ustawienie adresu pamięci
T	- Wykonanie ciągu instrukcji ze śledzeniem
U	- Wykonanie ciągu instrukcji bez śledzenia
X	- Wyprowadzenie stanu procesora

2.4. Opis zastosowania dyrektyw

Dyrektywa wprowadzania instrukcji

AS	- rozpoczęcie wprowadzania instrukcji od adresu S
-A	- usunięcie z pamięci modułu asembler/deasembler

UWAGA: Po zastosowaniu tej dyrektywy nie może być używana dyrektywa A, a dyrektywy L i T powodują wypisanie jedynie kodu instrukcji w postaci HEX.

S - początkowy adres wprowadzania, jeśli nie podany, to przyjmowany jest adres ostatnio ustawiony przez dyrektywy L, T lub U.

W instrukcjach można używać wyrażeń w dowolnej dozwolonej postaci.
Zakończenie wprowadzania po wciśnięciu znaku /CR/.

Np.

```
0100    MVI  C,A - .B
0102    LXI  H,.SOURCE
0105    LXI  D,+100    ; argument jest
                         równy.SOURCE
                         +100
0108    (CR)           ; znak kończący
                         asemblację
```

Dyrektywa wywołania programu

CS, B, D - wywołanie podprogramu od adresu S

gdzie:

S - adres startu podprogramu; podprogram musi kończyć się rozkazem RET

B - wartość wpisywana do pary rejestrów BC; jeśli nie jest podana, to wpisywane jest Ø.

D - wartość wpisywana do pary rejestrów DE, jeśli nie jest podana to wpisywane jest Ø.

np.

C.INITPRG, @ALFA, #12

Dyrektywy wyprowadzania zawartości pamięci

DS,F - wyprowadzenie zawartości pamięci w postaci bajtowej

DWS,F - jak wyżej w postaci słowowej

gdzie:

S - adres pierwszej wyprowadzanej komórki; gdy nie podana to przyjmowany jest adres ostatnio wyświetlony lub zawartość rejestrów HL, jeśli od czasu wejścia w ostatnią pułapkę nie wyświetlono zawartości pamięci.

F - końcowy adres wyprowadzania; jeśli nie podany, to wyprowadzanych jest 12 wierszy.

Dyrektywa w pierwszej postaci wyświetla bajty pamięci w następującym formacie:

AAAA BB BB BB ... BB CCCC ... C

gdzie:

AAAA - adres pierwszego wyprowadzonego bajtu w wierszu

BB BB... - ciąg /do 16 w wierszu/ dwucyfrowych liczb heksadecymalnych określający zawartość kolejnych bajtów. Pierwszy wyświetlany wiersz zawiera tyle liczb, aby początkowe adresy wyświetlania w następnych liniach były podzielne przez 16.

CCCC ... - Zawartość wyprowadzanych w wierszu bajtów przedstawiona w postaci znaków ASCII /do 16 znaków/.

Dyrektywa w drugiej postaci wyprowadza kolejne słowa /2 bajty/ pamięci w formacie:

AAAA WWWW WWWW WWWW ... WWWW CCCC ... CC

AAAA, CCC ... - jak w pierwszej postaci

WWWW - czterocyfrowe liczby HEX

Dyrektywa wpisywania do pamięci

FS, FD - wpisuje do pamięci od adresu S do F /włącznie/ stałą wartość określoną przez D

Dyrektywy uruchomienia programu

GP, A, B - uruchomienie programu od adresu P

-GP, A,B - jak wyżej; dyrektywa w drugiej postaci wypisuje kolejne słowa /2 bajty/ pamięci w formacie:

AAAA - przebieg nie osiągnie wartości 1

P - adres startu; gdy nie podany to start od bieżącej wartości PC,

A,B - adresy pułapek dynamicznych, po wejściu w którąkolwiek, wszystkie są likwidowane; gdy nie podane to pułapki nie są ustawiane.

Po wejściu w pułapkę jest wyświetlane:

    *A   .S

A - adres pułapki

S - symbol / o ile istnieje/ związany z tym adresem.

np.

```
   G100
 G.CRLF, .PRINT,  #1024
```

Dyrektywa obliczania i różnicy wyrażeń

H A, B - obliczenie i wyświetlenie /heksadecymalnie/ sumy i różnicy wyrażeń A i B,

H A - wyświetlenie wartości wyrażenia A w czterech postaciach heksadecymalnej, dziesiętnej, znakowej jako symbol /jeśli istnieje/

H - wyświetla całą listę symboli wraz z ich wartościami

np.

```
H 0, 1
H 'A' + = GAMMA
0042     #66 'B'
```

Dyrektywa wprowadzania wiersza

I CCCCCC ... Symulacja wprowadzania wiersza do programu

CCCC ... Dowolny ciąg znaków

np.

Jeśli program uruchamiany DUMP ma być później wywołany

DUMP X.COM

to w czasie testowania w celu wprowadzenia nazwy pliku X.COM należy wprowadzić:

IX.COM

Jeśli jako ciąg CCCC ... podane są nazwy 1 lub 2 plików, to są rezerwowane i inicjowane jeden lub dwa bloki opisu pliku FCB1 i FCB2 pod adresami odpowiednio 005C /FCB1/ i 006C /FCB2/.

/Uwaga: ponieważ pole FCB2 pokrywa częściowo pole FCB1, należy je przed użyciem przepisać w inny obszar/.

Dodatkowo linia CCCC ... jest wpisywana do bufora DBUFF /ADR.80-FF/

np.

J B: ALFA, ASM BETA.LIB

Dyrektywa J jest także używana w połączeniu z dyrektywą R w celu specyfikacji pliku.

Dyrektywa listowania pamięci

L S, F - listowanie w postaci mnemonicznej od adresu S do F z użyciem symboli

- L S, F - jak wyżej tylko bez symboli

gdzie:

S - adres pierwszej listowanej instrukcji; jeśli nie

podany to przyjmowany jest adres ostatnio listowany, translowany /Dyr. A/ śledzony lub ostatni adres pułapki /G,P/.

F - końcowy adres listowania; gdy nie podany to wyświetlane 12 wierszy

Format listowania:

SSSSSS:

AAAA   OPCODE   OPERAND   .TTTTTT

gdzie:

SSSSSS - symbol o wartości adresu AAAA /jeśli istnieje/

AAAA - adres instrukcji

OPCODE - kod instrukcji

OPERAND - argumenty instrukcji

TTTTTT - symbol /jeśli istnieje/ związany z argumentem.

Przy instrukcjach odwołujących się do pamięci jak np. INR M jest wypisywana wartość wskazywana przez argument OPCODE M = HH

Dyrektywa ustawiania pułapki

PP, C - ustawienie pułapki pod adresem P z licznikiem C

P - wyświetlenie wszystkich pułapek

-PP - likwidacja pułapki spod adresu P

- P   - likwidacja wszystkich pułapek

P   - adres pułapki

C   - początkowa wartość licznika pułapki; gdy nie podano to ustawione 1.

W jednym czasie może istnieć do 8 pułapek.

Każde wejście w pułapkę zmniejsza licznik o 1, jeśli nie jest on już równy 1.
Gdy równa się 1 to wykonywanie programu jest przerywane, po wykonaniu instrukcji gdzie ustawiana jest pułapka. Wejście w pułapkę nie likwiduje jej.

Każde wejście w pułapkę powoduje wyprowadzenie:

```
CC   PASS   PPPP   .SSSS
```

CC   - licznik pułapki

PPPP   - adres pułapki

SSSS   - symbol związany z tym adresem

Dyrektywa wczytania programu lub tabeli symboli

R   - wczytanie testowanego programu lub tablicy symboli z pliku wyspecyfikowanego w dyrektywie J,

RD   - jak wyżej, tylko dodaje wartość D do adresu ładowania

np.

```
JDUMP.COM   DUMP.SYM
R
```

Dyrektywa ustawiania adresu pamięci

SS - ustawienie bajtów począwszy od adresu S

SWS - ustawienie słów /dwóch bajtów/ począwszy od adresu S.

Dyrektywa powoduje wyprowadzenie wartości z pod adresu S, następnie oczekuje na wprowadzenie nowej wartości. Nowa wartość może byc podana w postaci wyrażenia lub dla S /wyświetlenie bajtów/ można wprowadzić ciąg znaków:

CCCCCC ....

Nowa wartość jest kończona znakiem /CR/.
Wprowadzenie samego znaku /CR/ nie zmienia zawartości i powoduje przejście do nowej komórki.
Znak "." kończy przeglądanie pamięci.

np.

```
S100
0100    34   C3   /CR/
0101    24    /CR/
0102    CF   "ASCII  /CR/
0103    6E   = X+5  /CR/
0108    D4    . /CR/
```

Dyrektywa wykonania N instrukcji ze śledzeniem

TN - wykonanie N instrukcji

- TN - jak wyżej; nie wyświetla symboli podczas śledzenia

TWN - wykonanie instrukcji bez śledzenia podprogramów

- TWN - jak wyżej; nie wyświetla symboli podczas śledzenia

gdzie:

N - liczba instrukcji do wykonania; gdy nie podana to przyjmowane 1

Dyrektywa wykonania N instrukcji

UN - wykonanie N instrukcji bez wypisywania stanu procesora

UWN - jak wyżej bez śledzenia podprogramów

gdzie:

N - liczba wykonywanych instrukcji; gdy nie podana to 1.

Dyrektywa jest realizowana analogicznie jak dyrektywa T, tylko bez wypisywania stanu procesora podczas wykonywania programu.
Wykonanie programu można w każdej chwili przerwać wciskając dowolny klawisz.

Dyrektywa wyprowadzająca wskaźniki i rejestry procesora.

X - wyprowadzenie zawartości wszystkich wskaźników i rejestrów procesora

XF - wyprowadzenie wartości podanego wskaźnika F i oczekiwanie na zmianę wartości

XR - wyprowadzenie zawartości rejestru /lub pary rejestrów/ R i oczekiwanie na zmianę zawartości

F - oznaczenie wskaźników procesora:

C - przeniesienie
Z - zero
M - minus
E - parzystość
I - przeniesienie połówkowe

R - oznaczenie rejestrów lub pary rejestrów:

A - akumulator
B - para BC
D - para DE
H - para HL
S - wskaźnik stosu SP
P - licznik rozkazów PC

Wskaźniki i rejestry są wyświetlane w postaci:

CZMEJ A=HH B=HHHH D=HHHH S:HHHH P=HHH OPCODE SYM

CZMEJ - Wartości wskaźników: odpowiednia litera, gdy wskaźnik ustawiony - /kreska/ gdy wyzerowany,

HHHH - Zawartość rejestrów w postaci liczby heksadecymalnej

OPCODE - Instrukcja w postaci mnemonicznej

SYM - Argument instrukcji w postaci symbolu

np.

X

C---J A=23 B=0000 D=23FF H=5001 S=FF00 P=1245 LXI H,2001
TAB XB

B=000 1234 /CR/.

Druk: ZE „ELWRO", MGW--zam.484/87/2 000